USING PARALLEL PARTICLE FILTERS FOR INFERENCES IN HIDDEN MARKOV MODELS

HENG CHIANG WEE

(M. Sc, NUS)

SUPERVISORS: PROFESSOR CHAN HOCK PENG & ASSOCIATE PROFESSOR AJAY JASRA

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN STATISTICS

DEPARTMENT OF STATISTICS AND APPLIED PROBABILITY

NATIONAL UNIVERSITY OF SINGAPORE

2014

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Heng Chiang Wee

19 December 2014

Acknowledgements

Firstly, I would like to thank my supervisors for their guidance and patience during the writing and development of this thesis. They have been supportive and understanding of my work commitments and have provided valuable advice to me. I feel that I have become more matured through my interactions with them.

I would also like to express my sincere thanks to Mr Kwek Hiok Chuang, the principal of Nanyang Junior College, for kindly endorsing my professional development leave. As a teacher, I need to spend time with my students and guide them in their studies. Thanks to the additional time that was granted to me, I was able to complete the writing of this thesis. To my department head Mrs Lim Choy Fung, I would like to thank her for supporting my decision to pursue my PhD and making arrangements for someone to take over my classes whenever necessary. The department as a whole has been supportive and understanding of my commitment to finish this PhD, and rendered assistance whenever asked.

To my mother, who left me and my sister recently, I would like to tell her that she had always been my beacon and inspiration. That her love and care for me over the years, her support of me and encouragements when things are not going well, have made me into a better person. I will always remember her.

I would like to go on to thank Loo Chin for standing by me through all these years. She has given me two adorable children Eleanor and Lucas, and brought them up well (and a third one is on the way). I think the coming years will be a busy period for the two of us, but whenever we look at them, seeing their innocence and playfulness, we know that any sacrifice is worth it.

Finally, this thesis is dedicated to all who have helped me in one way or another.

Contents

Summary

In this thesis, we use particle filters on segmentations of the latent-state sequence of a hidden Markov model, to estimate the model likelihood and distribution of the hidden states. Under this set-up, the latent-state sequence is partitioned into subsequences, and particle filters are applied to provide estimation for the entire sequence. An important advantage is that parallel processing can be employed to reduce wall-clock computation time. We use a martingale difference argument to show that the model likelihood estimate is unbiased. We show, on numerical studies, that the estimators using parallel particle filters have comparable or reduced (for smoothed hidden-state estimation) variances compared to those obtained from standard particle filters with no sequence segmentation. We also illustrate the use of the parallel particle filter framework in the context of particle MCMC, on a stochastic volatility model.

List of Tables

List of Figures

Author's Contribution

In this thesis, the author has proposed a new method for the estimation of the likelihood function and latent states distributions in a hidden Markov model. The main idea is to use segmentation of the observed data and run particle filters for each segment in parallel. The framework and notations used for this proposed method are introduced by the author. The author has proposed an estimate for $\psi_{MT} := \mathbb{E}[\psi(X_{1:MT})|Y_{1:MT}]$ under this framework and proved that it is unbiased in Theorem 3.4. The proof is motivated by Chan and Lai (2013) where a martingale difference approach is used to prove the unbiasedness of the estimate using the proposed method. The author considered two martingale difference expressions for the estimates for the proof of unbiasedness. The validity of the expressions are proven by the author in Lemmas 3.3 and 3.6. Technical lemmas meant for these proofs are done by the author as well. The two different martingale difference expressions can be used for establishing a central limit theorem and standard errors estimates respectively. A discussion on the possible computational cost savings is done by the author.

The author has conducted numerical studies to support the validity of the proposed method that can be found in Chapter 4. The proposed method was used to compute likelihood estimates for a Gaussian linear hidden Markov model. Comparisons are made to the usual particle filter and Kalman filtering to assess the performance of

the proposed method in likelihood estimation. Further, simulations are done by the author to investigate the possible variance reduction in smoothed means estimation using the proposed method.

The author has further done numerical studies on real-life financial data using a stochastic volatility model used by Flurry and Shephard (2011) in Chapter 5. In this study, the proposed method is used in conjunction with the particle Markov chain Monte Carlo method that was proposed by Andrieu *et al.* (2010). The use of particle Markov chain Monte Carlo method in econometrics has been well reviewed by Pitt (2012). The numerical study illustrates the use of the proposed method in existing Markov chain Monte Carlo methods and minimal adjustments are required for this purpose.

Chapter 1

Introduction

In this thesis, our objective is to propose a new sequential Monte Carlo (SMC) method for the estimation of the likelihood function and latent states in a hidden Markov model. A hidden Markov model (HMM) is a class of models with a wide range of real applications. The application of this model includes speech recognition, econometrics and computational biology. For this model, in contrast to the simple Markov model, the Markov chain is not directly observed and hence the use of the adjective 'hidden' to describe this chain. The observation is done, typically with noise, via an output that is dependent on the current state of the Markov chain. For such model, one will be interested to obtain the distribution of the hidden (latent) states conditioned on the observations gathered. Having this distribution will enable one to make inferences of the sequence of states for the hidden Markov chain. Another quantity of interest is the likelihood function for the observations. Apart from obtaining the probability for the sequence of observations, the likelihood function is useful for model comparison and parameter estimation. However, the exact computation of the required conditional distributions and the likelihood function is typically hard to compute due to the fact that the associated high-dimensional integrals are difficult to evaluate. In practice, one will make use of numerical methods to approximate the conditional distributions and the likelihood function.

A common and popular method to obtain these estimates is to use SMC methods,

otherwise known as particle filtering. The details will be reviewed in Chapter 2. As a brief introduction, the algorithm will generate samples known as particles with assigned weightings. Each particle consists of a string of simulated random variables for the hidden states with respect to time. These weightings are computed to ensure that the particles provide an approximation to the target density of the hidden Markov model. One of the advantages of SMC methods is that the particles can be updated immediately as new observations are available as time progresses. Typical SMC methods will have a resampling stage to address the problem of weight degeneracy of the particles as time propagates. However, with resampling, one will expect less distinct particles in the earlier stages of time, a problem known as *path degeneracy*.

The motivation behind our proposed method is driven by parallel computation. With recent developments in computation processors, parallel computing has gained attention for its computational cost savings advantage, as algorithms can be run in tandem. Our aim is to utilise this facility for the implementation of the proposed algorithm by considering data segmentation of the observation sequence of a hidden Markov model. The details will be dealt with in Chapter 3. Apart from computational cost savings, our proposed method has other attractive properties as well. Our proposed method will be able to tackle the problem of path degeneracy. One possible advantage of solving the path degeneracy problem is variance reduction for the estimation of smoothed means for the hidden Markov model. The terminologies for the notations mentioned will be addressed in Chapter 2.

For this thesis, we are also interested in Bayesian inferences from a hidden Markov model, where the parameters involved are given a prior distribution. Under this framework, inferences of the parameter are based on the posterior distribution (which will be defined later) of the parameter. We shall provide a brief review of Bayesian inference problems in this chapter. It is explained that the analytical solutions to Bayesian problems are often intractable and thus necessitate the need of a numerical method to obtain approximated solutions. The common numerical approach is to make use of Markov chain Monte Carlo (MCMC) methods to target the density involved for the

Bayesian problem, the posterior density. Recent developments in this area utilise a particle filter to obtain unbiased estimates of the likelihood function. These estimates are used for the computation of the acceptance ratio of the MCMC methods. One such method is the particle Markov chain Monte Carlo (PMCMC) method proposed by Andrieu *et al.* (2010). Our proposed method can replace the existing usage of particle filters for the estimation of the likelihood function. Using our proposed method will enable one to achieve computational cost savings with minimal adjustment to the existing PMCMC algorithm.

Accordingly, in this chapter, we will first provide a quick review on Bayesian inferences in Section 1.1. In Section 1.2, we will give the definition of conditional expectations and relevant properties that will be used for proving our main theorem in Chapter 3. The thesis organisation will be provided in Section 1.3. In this section, we will provide the reader an overview on the structure of the thesis. The organisation can be summarised to the following three areas: literature review; notations, framework and theory of the proposed method; and numerical studies on selected hidden Markov model. The details will be provided in this section.

1.1 Review on Bayesian inferences

In classical statistical theory, parameter inferences are often done using a *maximum likelihood method*. Before we introduce the Bayesian paradigm, we shall give a quick recap of this method. More details can be found in, for example, Shao (2003).

Definition 1.1.1. Let $X \in \mathcal{X}$ be a sample with a probability density function f_θ with respect to a σ-finite measure ν, where $\theta \in \Theta \subset \mathbb{R}^k$.

(i) For each $x \in \mathcal{X}$, $f_\theta(x)$ considered as a function of θ is called the *likelihood function* and denoted by $\ell(\theta)$.

(ii) Let $\bar{\Theta}$ be the closure of Θ. A $\hat{\theta} \in \bar{\Theta}$ satisfying $\ell(\hat{\theta}) = \max_{\theta \in \bar{\Theta}} \ell(\theta)$ is called a

maximum likelihood estimate (MLE) of θ. If $\hat{\theta}$ is a Borel function of X a.e. ν, then $\hat{\theta}$ is called a *maximum likelihood estimator* (MLE) of θ.

(iii) Let g be a Borel function from Θ to $\mathbb{R}^p$, $p \leq k$. If $\hat{\theta}$ is an MLE of θ, then $\hat{\vartheta} := g(\hat{\theta})$ is defined to be an MLE of $\vartheta := g(\theta)$.

The MLE of θ can often be found using the *likelihood equation*

$$\frac{\partial \ell(\theta)}{\partial \theta} = 0.$$

In the Bayesian inferential framework, the parameters of the likelihood function are treated as random variables. When no data is observed, the parameter θ is given a distribution called the *prior distribution* with density $\pi(\theta)$. The information is provided by the data x, which is an observation from X with a distribution parameterised by θ with density $f(x|\theta)$. As the data is observed, we will be interested in the *posterior distribution* with density $\pi(\theta|x)$ to make inferences about the unknown θ conditioned on the observed data x. The posterior density can be computed using *Bayes Theorem*:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}. \tag{1.1.1}$$

When equation (1.1.1) is used in the evaluation of conditional density, we can compute the posterior density by

$$\pi(\theta|x) = \pi(\theta) f(x|\theta)/m(x), \tag{1.1.2}$$

where $m(x) := \int \pi(\theta) f(x|\theta)\, d\theta$ is the *marginal density* of X.

In the Bayesian approach, all the information about θ is provided from the posterior distribution. Accordingly, inferences about θ must be made from the posterior distribution. In estimating θ, the decision-theoretic approach is to specify a *loss function* $L(\theta, \delta(x))$ which denotes the loss incurred when $\delta(x)$ is used to estimate θ. The *Bayes risk* is the expectation of the loss function with respect to the posterior distribution given by

$$\mathbb{E}[L(\theta, \delta(x))|X = x] = \int L(\theta, \delta(x)) \pi(\theta|x)\, d\theta.$$

The *Bayes action* is the decision that minimises the Bayes risk. Let $\|f\|^2 = \left(\int f^2\, d\mu\right)^{\frac{1}{2}}$ denotes the L^2-norm. For the quadratic loss function $L(\theta, \delta(x)) = \|\theta - \delta(x)\|^2$, the

Bayes action is $\delta^\pi := \mathbb{E}_\pi[\theta|x]$ where the expectation is taken under the posterior distribution. In contrast, the maximum likelihood method does not typically make use of any loss function.

In order to evaluate the Bayes action under the quadratic loss function, one will need to have the posterior density in closed form. This is often not possible unless the prior is a *conjugate prior*. When a conjugate prior is chosen, the prior and posterior distributions will belong to the same parametric family of distributions or a pair of parametric families. The parametric families are often exponential which allows explicit computation and updating of the parameters involved. However, even when the posterior density is in closed form, it may not be possible to evaluate the integral $\int \theta\pi(\theta|x)\, d\theta$ analytically. For such situations, a numerical method is necessary to approximate the integrals involved. One would need to make use of sampling techniques to produce approximate samples from the posterior distribution for instance.

In Chapter 2, we will give a brief review on various sampling techniques that are popular in practice. We will review some recent developments in these sampling techniques. These techniques provide a basis for our proposed method.

1.2 Conditional expectations and martingales

The main theorem in this thesis is proven using *conditional expectation* and *difference of martingale sequences*. As such, we shall provide a quick review of conditional expectation and martingale in this section. The reader can refer to Billingsley (1995), Durrett (1995) and Chung (2001) for a more detailed treatment of these topics.

Definition 1.2.1. Consider a probability space $(\Omega, \mathcal{F}_0, P)$, a σ-field $\mathcal{F} \subset \mathcal{F}_0$, and a random variable X that is measurable with respect to $\mathcal{F}_0$ with $\mathbb{E}[|X|] < \infty$. The *conditional expectation* of X given $\mathcal{F}$, $\mathbb{E}[X|\mathcal{F}]$, is a random variable Y such that

(i) Y is measurable with respect to $\mathcal{F}$,

(ii) for all $A \in \mathcal{F}$, $\int_A X\, dP = \int_A Y\, dP$.

We shall list down some useful properties of conditional expectations that will be used in the proof of Theorem 3.4 and Theorem 3.6.

Theorem 1.1

The conditional expectation satisfies the following properties.

(i) Linearity: $\mathbb{E}[aX + bY|\mathcal{F}] = a\mathbb{E}[X|\mathcal{F}] + b\mathbb{E}[Y|\mathcal{F}]$.

(ii) Monotonicity: If $X \leq Y$, *then* $\mathbb{E}[X|\mathcal{F}] \leq \mathbb{E}[Y|\mathcal{F}]$ *a.s.*

(iii) Tower Property: If $\mathcal{F}_1 \subset \mathcal{F}_2$, *then (a)* $\mathbb{E}[\mathbb{E}[X|\mathcal{F}_1]|\mathcal{F}_2] = \mathbb{E}[X|\mathcal{F}_1]$ *and (b)* $\mathbb{E}[\mathbb{E}[X|\mathcal{F}_2]|\mathcal{F}_1] = \mathbb{E}[X|\mathcal{F}_1]$.

(iv) If $X \in \mathcal{F}$ *and* $\mathbb{E}[|Y|] < \infty$, $E[|XY|] < \infty$, *then* $\mathbb{E}[XY|\mathcal{F}] = X\mathbb{E}[Y|\mathcal{F}]$ *a.s.*

To introduce a martingale, one will need to consider a sequence of σ-algebras $\{\mathcal{F}_n\}_{n=1}^{\infty}$ in $\mathcal{F}$ such that $\mathcal{F}_n \subset \mathcal{F}_{n+1}$ for all $n \geq 1$. For such sequences, we say that $\{\mathcal{F}_n\}_{n=1}^{\infty}$ is a *filtration*.

Definition 1.2.2. Let $X_1, X_2, \ldots$ be a sequence of random variables on a probability space $(\Omega, \mathcal{F}, P)$ and let $\{\mathcal{F}_n\}_{n=1}^{\infty}$ be a filtration in $\mathcal{F}$. Let X_n be measurable with respect to $\mathcal{F}_n$ for $n \geq 1$ with $\mathbb{E}|X_n| < \infty$. The sequence $\{(X_n, \mathcal{F}_n) : n = 1, 2, \ldots\}$ is a *martingale* if $\mathbb{E}[X_{n+1}|\mathcal{F}_n] = X_n$ a.s.

Condition (iv) implies that $\mathbb{E}[X_m|\mathcal{F}_n] = X_n$ a.s for $m > n$. This can be proven easily using induction and the tower property.

1.3 Thesis organisation

The thesis is organised in the following manner.

In Chapter 2, we will provide a literature review on various sampling methods that are popular in practice. In particular, the SMC method will be discussed in greater depth. A brief account on Markov chains will be provided in this chapter, as well as a review on Markov chain Monte Carlo (MCMC) methods. Recent developments of the MCMC methods will also be discussed. We shall consider the particle Markov chain Monte Carlo method, SMC^2 algorithm and substitution algorithm for this purpose.

In Chapter 3, the notations and framework of the proposed method will be introduced. The method, termed as the *parallel particle filter* (PPF) will be elaborated on. The algorithm of PPF will be given in this chapter. The estimates of the marginal likelihood and functions involving latent states using PPF framework will be defined. The proof of the unbiasedness property of the proposed estimates under the canonical case will be given. The proof will make use of a martingale difference expression of the proposed estimates. We shall give two martingale difference expressions for the proposed estimates. While both can be used to prove unbiasedness, these two expressions serve different purposes. One form is useful in obtaining a central limit theorem for the proposed estimate; while the other form is useful in deriving approximation for the standard errors of the proposed estimates. A brief discussion on computational cost savings when the PPF is used will be done.

In Chapter 4, we will conduct a numerical study for using PPF to estimate the marginal likelihood and smoothed mean for a chosen linear Gaussian HMM. We will compare the performance of PPF with Kalman filtering and the usual particle filter (PF) via estimates of the marginal likelihood and latent states. We will further discuss the implementation of the PPF algorithm when different number of subsequences are used. A brief account on using subsampling for our proposed method will be given.

In Chapter 5, before we proceed to the numerical study on real data for a stochastic volatility (SV) model, a brief review will be given for selected SV models. The SV model chosen for our numerical study is proposed by Flury and Shephard (2011). We will make some modification to the model and employ PMCMC algorithm util-

ising PPF for Bayesian inferences. Autocorrelation plots will be used to assess the performance of the MCMC method with the implementation of our proposed method.

Concluding remarks and discussion on future research will be given in Chapter 6. In particular, we will look into proving a central limit theorem for our proposed estimates and obtaining approximations for the standard errors of our proposed estimates.

Chapter 2

Literature Review

In this chapter, we will review important topics used in this thesis. The structure of this chapter can be categorised as follows: definitions of hidden Markov model, sampling techniques leading to sequential Monte Carlo methods, and Markov chain Monte Carlo methods and their extensions. We will elaborate on the topics covered for each section.

As our proposed method is applied to the hidden Markov model, we will provide the definition of this model in Section 2.1, as well as an algorithm to compute the exact distribution under certain conditions. Thereafter, we will introduce various numerical methods that are widely used for simulation and estimation purposes. We shall begin by first reviewing the Monte Carlo method in Section 2.2 and importance sampling in Section 2.3. These methods are fundamental to various sampling techniques. An example of such extension is the self-normalised importance resampling which will be discussed in Section 2.4. Another important extension in the context of our thesis is sequential Monte Carlo (SMC) methods which will be covered in Section 2.5. When applied to hidden Markov models, SMC methods are commonly known as particle filtering. We will give a detailed review of SMC methods in this section as our proposed method is an example of such algorithms. We shall provide the notations and fundamentals of particle filters to prepare the reader for the discussion in Chapter 3. In this thesis, we are interested in Bayesian inferences involving a hidden Markov

model: the parameters involved are given a prior distribution and one has to make inferences from the posterior density. A popular approach is to make use of Markov chain Monte Carlo (MCMC) methods for such scenarios. Accordingly, we shall first give a quick review on the convergence of a Markov chain to its stationary distribution if it exists before discussing MCMC methods in Section 2.6. In particular, we will consider two widely used MCMC methods: the Metropolis-Hastings (MH) algorithm and Gibbs sampling. These methods, apart from being easy to implement for simple cases, can serve as building blocks for more complex algorithms. In our thesis, we shall consider a MH within Gibbs sampling for our numerical studies in Chapters 5. As we will show, one needs the exact marginal likelihood for a hidden Markov model to compute the MH acceptance ratio in a MH algorithm. When the exact marginal likelihood is intractable, one could make use of unbiased estimates to compute the required MH acceptance ratio. These algorithms are examples of pseudo Markov chain Monte Carlo (PsMCMC) methods (proposed by Beaumont (2003) and Andrieu and Roberts (2009)) and will be discussed briefly in Section 2.7. The principles of PsMCMC methods are fundamental to the *particle Markov chain Monte Carlo (PMCMC)* methods (proposed by Andrieu *et al.*, 2010) that will be discussed in Section 2.8 and SMC^2 methods (proposed by Chopin *et al.*, 2013) that will be discussed in Section 2.9. Finally in Section 2.10, we will introduce the *substitution algorithm* that was proposed by Chan and Lai (2014) as a new alternative to the usual MCMC methods.

2.1 Hidden Markov model

The hidden Markov model (HMM) or the state-space model (SSM) is a class of statistical models that have a wide variety of real applications. The use of hidden states in this model allows it to model many real world time series. We shall list some of its uses. Rabiner and Juang (1993), and Jelinek (1997) made use of HMM in speech recognition models. In econometrics, Hamilton (1989) and Kim and Nelson (1999) utilised HMMs in financial models. HMMs are used in computational biology as well. Interested readers can refer to Durbin *et al.* (1998) and Koski (2001) and references

therein for an in-depth treatment. Other examples where HMMs are used include computer vision (Bunke and Caelli, 2001), information theory (Elliott, 1993) and location tracking (Gordon *et al.* (1993), Ristic *et al.* (2004)). For descriptions of the models used in real world time series involving HMMs, interested readers can refer to Cappé (2005). In this section, we will provide a brief introduction of this model and the notations associated with such models.

Definition 2.1.1. A *hidden Markov model* comprises of a hidden Markov state process $\{X_n : n \geq 1\}$ described by its initial density $X_1 \sim \mu_\theta(\cdot)$ and transition probability density $X_{n+1}|(X_n = x) \sim f_\theta(\cdot|x)$ for some static parameter $\theta \in \Theta \subseteq \mathbb{R}^d$ and the observed process $\{Y_n : n \geq 1\}$ which is related to the Markov state process through the density $Y_n|(X_n = x) \sim g_\theta(\cdot|x)$. We can also assume that θ is a variable and assign a prior $p(\theta)$ to it.

In our studies of the HMM, we are interested in two main areas: the estimation of the hidden (latent) states and the estimation of the parameters involved. For state inference when θ is fixed, there are three main areas of interest. If the process $\{Y_n\}$ is observed, the estimation of the density of X_k for $k < n$, $p_\theta(x_k|y_{1:n})$, is known as *smoothing*; the estimation of the density of X_n, $p_\theta(x_n|y_{1:n})$, is known as *filtering* and the estimation of the density of X_k, $p_\theta(x_k|y_{1:n})$, for $k > n$ is known as *predicting*, where $y_{1:n}$ denotes the observations $y_1, \ldots, y_n$. In this thesis, we will focus our attention on filtering and estimation of smoothed mean $\mathbb{E}[X_k|Y_{1:n}]$ for $k < n$, where $\mathbb{E}[\cdot]$ denotes the expectation taken under the HMM.

A classic example of an HMM is the Gaussian linear state-space model. It takes the form

$$\begin{aligned} X_{k+1} &= A_k X_k + R_k U_k, \\ Y_k &= B_k X_k + S_k V_k, \end{aligned}$$

where $\{U_k\}_{k\geq 0}$, the state or process noise, and $\{V_k\}_{k\geq 0}$, the measurement noise, are independent standard multivariate Gaussian white noise, R_k is the square root of the

Algorithm 1: Kalman Filtering

begin

Initialisation;

Set $\hat{X}_{0|0} = 0$ and $\Sigma_{0|0} = \Sigma_\nu$;

for $k = 1 : n$ **do**

Compute the following:

$$\begin{aligned}
\hat{X}_{k|k-1} &= A_k \hat{X}_{k-1|k-1}, && \text{Prediction} \\
\Sigma_{k|k-1} &= A_k \Sigma_{k-1|k-1} A_k' + R_k R_k', && \text{Prediction} \\
\epsilon_k &= Y_k - B_k \hat{X}_{k|k-1}, && \text{innovation} \\
\Gamma_k &= B_k \Sigma_{k|k-1} B_k' + S_k S_k', && \text{innovation covariance} \\
K_k &= \Sigma_{k|k-1} B_k' \Gamma_k^{-1}, && \text{Kalman gain} \\
\hat{X}_{k|k} &= \hat{X}_{k|k-1} + K_k \epsilon_k, && \text{filter state estimation} \\
\Sigma_{k|k} &= \Sigma_{k|k-1} - K_k B_k \Sigma_{k|k-1}, && \text{filter error covariance.}
\end{aligned}$$

end

end

state noise covariance and S_k is the square root of the measurement noise covariance. A_k and B_k are known matrices with appropriate dimensions and dependent on the time index k. The initial condition X_0 is Gaussian with mean 0 and covariance Σ_ν and is uncorrelated with the processes $\{U_k\}$ and $\{V_k\}$.

This particular model is important in engineering and time series due to its practical application. Further, it is one of the model where the distribution of X_n given Y_n can be computed using an exact numerical algorithm. The algorithm is known as *Kalman filtering* which is introduced by Kalman and Bucy (1961). The pseudo code is given in Algorithm 1.

For general HMMs, typically we will not be able to have an algorithm to obtain the exact distribution of the hidden states given the observed states. For example,

consider the following HMM

$$X_{k+1} = a(X_k) + \epsilon_k, \quad Y_k = b(X_k) + \nu_k$$

where $a(\cdot)$ and $b(\cdot)$ are measurable functions and $\{\epsilon_k\}_{k\geq 0}$ and $\{\nu_k\}_{k\geq 0}$ are mutually independent and identically distributed sequences of random variables that are independent of X_0. If $a(\cdot)$ and $b(\cdot)$ are non-linear, one will need to use other methods to obtain approximation of the required distribution of X_n given $Y_{1:n}$ or other values of interest.

2.2 Monte Carlo method

Suppose one is interested in approximating the expected value of a function of random variable X taking values on the space $\mathcal{X}$ with probability measure P, denoted by $\mu(f) := \mathbb{E}_P[f(X)]$ where $f : \mathcal{X} \mapsto \mathbb{R}$ is such that $\mu(f) < \infty$. One could use the technique of *Monte Carlo* method to obtain an unbiased estimate of $\mu(f)$. The idea is as follows. If one is able to generate a sample $(X_1, \ldots, X_N)$ directly from the distribution P, then one can obtain an estimate, called the *Monte Carlo estimate*, using the empirical average given by

$$\hat{\mu}(f) = \frac{1}{N}\sum_{k=1}^{N} f(X_k). \tag{2.2.1}$$

It is a routine exercise to show that (2.2.1) gives an unbiased estimate for $\mu(f)$. By the Strong Law of Large Numbers, $\hat{\mu}(f)$ converges almost surely to $\mu(f)$.

Apart from its simplicity, one will be able to determine the speed of convergence of the Monte Carlo estimate if $\mathbb{E}_P\{[f(X)]^2\}$ is finite. Define

$$\sigma^2 := \mathbb{V}_P[f(X)] = \mathbb{E}_P\{[f(X)]^2\} - \{\mathbb{E}_P[f(X)]\}^2.$$

The variance of the Monte Carlo estimate is given by σ^2/N. Accordingly, the standard error of the Monte Carlo estimate will decrease at a rate of $\mathcal{O}(1/\sqrt{N})$ regardless

of the dimension of the space $\mathcal{X}$.

Further, since $\sigma^2 < \infty$, one will be able to establish a Central Limit Theorem for the Monte Carlo estimate given by

$$\frac{\sqrt{N}(\hat{\mu}(f) - \mu(f))}{\sigma} \Rightarrow N(0,\, 1)$$

where $\Rightarrow$ denotes convergence in distribution. Hence, for large N,

$$\hat{\mu}(f) \approx \mu(f) + \frac{\sigma}{\sqrt{N}} Z$$

in distribution where $Z \sim N(0,\, 1)$. Accordingly, the convergence rate of the Monte Carlo estimate is $\mathcal{O}(1/\sqrt{N})$.

One can approximate σ^2 by

$$\hat{\sigma}^2 := \frac{1}{N} \sum_{k=1}^{N} \left[f(X_k) - \hat{\mu}(f) \right]^2 .$$

Applying the Strong Law of Large Numbers, one can show that $\hat{\sigma}^2$ converges almost surely to σ^2. Using this estimate and the Central Limit Theorem, one would be able to construct asymptotic confidence bounds for the Monte Carlo estimates.

2.3 Importance sampling

As in the earlier section, suppose one is interested in approximating $\mu(f)$. Instead of sampling X directly with respect to probability measure P, one could use the technique of *importance sampling* (IS) to generate the random variable with respect to another probability measure Q such that $P \ll Q$. This is done in the case when it is simpler to generate with respect to the probability measure Q. Further, this technique may result in variance reduction for the estimates involved.

As an example of variance reduction, one can consider the estimation of

$$\alpha = \mathbb{E}_P[\mathbb{I}_{(k,\infty)}(X)] = \mathbb{P}(X > k)$$

where α is small and $\mathbb{I}_A(x)$ is the indicator function of the set A. The idea is to simulate from another distribution such that the event $\{X > k\}$ will occur with a higher probability. Under this procedure, the 'important' values are given higher weighting. We will then adjust it with a weight given by the Radon-Nikodym derivative $\frac{dP}{dQ}$ of the two distributions involved, which is termed as *importance weight*. The estimate is given by

$$\hat{\alpha}_{IS} := \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}_{(k,\infty)}(X_i)\frac{dP}{dQ}(X_i).$$

Since $\{X_1, \ldots, X_N\}$ is generated from the distribution Q, one will be able to show that $\hat{\alpha}_{IS}$ is unbiased for α. Further, the variance of the importance sampling estimate is given by

$$\mathbb{V}[\hat{\alpha}_{IS}] = \frac{1}{N}\left\{\mathbb{E}_Q\left[\left(\frac{dP}{dQ}(X)\right)^2 \mathbb{I}_{(k,\infty)}(X)\right] - \alpha^2\right\}.$$

For the Monte Carlo estimate of α given by $\hat{\alpha}_{MC} := N^{-1}\sum_{i=1}^{N} \mathbb{I}_{(k,\infty)}(X_i)$, the variance of this estimate is given by

$$\mathbb{V}[\hat{\alpha}_{MC}] = \frac{\alpha(1-\alpha)}{N}.$$

Accordingly, if we choose Q such that $\mathbb{E}_Q\left[\left(\frac{dP}{dQ}(X)\right)^2 \mathbb{I}_{(k,\infty)}(X)\right] < \alpha$, one will be able to obtain an estimate with a lower variance. Since the variance of the estimator is reduced, one will have a more efficient estimate.

More generally, if one requires estimate of $\mu(f)$, one can perform importance sampling to obtain a sample $\{X_i\}_{i=1}^N$ with associated importance weights $\{\frac{dP}{dQ}(X_i)\}_{i=1}^N$. Then the importance sampling estimate of μ is given by

$$\hat{\mu}_{IS}(f) := \frac{1}{N}\sum_{i=1}^{N} f(X_i)\frac{dP}{dQ}(X_i). \tag{2.3.1}$$

Since $\mathbb{E}_Q[f(X)\frac{dP}{dQ}(X)] = \mathbb{E}_P[f(X)]$, one can show that the importance sampling estimate is unbiased. Accordingly, by the Strong Law of Large Numbers, the IS estimates will be consistent as well.

For this technique, the probability measures P and Q must be known exactly. For probability measures that are known up to a constant, one can make use of *self-*

normalised importance resampling which will be introduced in the next section.

2.4 Self-normalised importance sampling

From the earlier section, for importance sampling to work, one would need to know P and Q exactly. To elaborate on this, suppose P is known up to a constant multiplier. That is, the density function can be expressed as $p(x) = bp_u(x)$ where b is an unknown constant and p_u is the unnormalised density function which is known exactly. Further, the proposal density can be expressed as $q(x) = cq_u(x)$ where c is an unknown constant and q_u is the unnormalised density function which is known exactly. Under this scenario, the importance weights can be expressed as

$$w(X_k) := \frac{dP}{dQ}(X_k) = \frac{(b/c)p_u(X_k)}{q_u(X_k)}$$

where b/c is an unknown constant. Apart from the case when $b = c$, the importance weights, having an unknown component cannot be evaluated. For such scenario, one will not be able to use importance sampling.

To circumvent this problem, one can consider the *self-normalised importance sampling (SIS) estimate* given by

$$\tilde{\mu}_{IS}(f) = \frac{\sum_{k=1}^{N} f(X_k)w(X_k)}{\sum_{k=1}^{N} w(X_k)}. \tag{2.4.1}$$

Since $w(X_k)$ appears in both the numerator and denominator of (2.4.1), the term b/c cancels and one will be able to evaluate the estimate. By doing so, we have normalised the importance weights, resulting in the estimate to be self-normalising. One can define the normalised importance weights by

$$W(X_k) := \frac{w(X_k)}{\sum_{m=1}^{N} w(X_m)}$$

and the SIS estimate can be written as

$$\tilde{\mu}_{IS}(f) := \sum_{k=1}^{N} f(X_k)W(X_k).$$

However, by normalising the importance weights, one will not be able to obtain an unbiased estimate. Since $\mathbb{E}_Q[f(X_k)W(X_k)]$ is not equal to $\mu(f)$ in general, the SIS estimate will be biased. Despite this, one can show that the SIS estimate is consistent for $\mu(f)$ under certain conditions to be specified in the following theorem.

Theorem 2.1

Let p be a probability density function on $\mathbb{R}^d$ of the measure P and let $f(x)$ be a function such that $\mu(f) = \mathbb{E}_P[f(X)]$ exists. Suppose that q is a probability density function on $\mathbb{R}^d$ with $q(x) > 0$ whenever $p(x) > 0$. Let $X_k \sim Q$, $k = 1, \ldots, N$ be independent. Then the SIS estimate satisfies

$$\mathbb{P}\left(\lim_{N\to\infty} \tilde{\mu}_{IS}(f) = \mu(f)\right) = 1.$$

Proof:

One can first express the SIS estimate as

$$\tilde{\mu}_{IS}(f) = \frac{\frac{1}{N}\sum_{k=1}^{N} f(X_k)w(X_k)}{\frac{1}{N}\sum_{k=1}^{N} w(X_k)}.$$

The numerator is the usual IS estimate while the denominator is the IS estimate setting $f \equiv 1$. Using the Strong Law of Large Numbers, we have

$$\frac{1}{N}\sum_{k=1}^{N} f(X_k)w(X_k) \to \mu(f) \; a.s. \quad \text{and} \quad \frac{1}{N}\sum_{k=1}^{N} w(X_k) \to 1 \; a.s.$$

Consequently, $\tilde{\mu}_{IS} \to \mu(f)$ a.s. □

Theorem 2.1 justifies the use of the SIS estimate when the density functions p and q are known up to constant multipliers. In the next section, we will discuss the use of importance sampling when the distribution P is sequential in nature. This class of methods are known as sequential Monte Carlo methods. The idea of normalising the weights will be used in these algorithms in order to deal with the normalising constant.

2.5 Sequential Monte Carlo methods

In the previous section, we have described tweaking the importance sampling technique to target a distribution P when it is known up to a constant. In such scenarios, the target distribution is of a fixed dimension. In some situations, however, one is interested to obtain samples from a sequence of distributions $\{\pi_t\}$ where the dimension of π_t increases as t increases. For such scenarios, the parameter t is often related to time. If one is to generate N samples for each distribution π_t for $t \geq 1$ when N is large, one would need more storage space for the samples (which are increasing in dimension with t). Further, there might be more computational complexity to target π_t directly as t increases. If one is faced with a time or storage constraint to store the generated samples as t increases, one will not be able to use the usual importance sampling techniques to target this sequence of distributions.

To target such sequence of distributions, one could make use of sequential Monte Carlo (SMC) methods. SMC methods refer to algorithms that sample sequentially from a sequence of target distribution $\{\pi_n\}$ of increasing dimension, where each distribution π_n is defined on the product space $\mathcal{X}^n$. In these scenarios, it is often that the state space of π_{t+1} is an augmentation of the state space of π_t and one could simply simulate a subvector for the approximation of π_{t+1}. One such example, that was used by Gordon *et al.* (1993), is the position and speed of a plane at time t where the observations are obtained with noise.

A natural example of using SMC methods is in HMMs due to its sequential nature with respect to time. For an HMM, when a new observation is recorded, the importance weights can be updated sequentially by choosing appropriate sampling distributions. This avoids the inefficiency and trouble of regenerating the entire samples. Accordingly, for this section, we will focus our attention on applying the SMC method to a hidden Markov model. In the HMM context, SMC methods are also referred to as *particle filtering*. We will discuss the operations of these algorithms in the following subsections.

Throughout this section, we shall consider the following hidden Markov model for our discussion:

$$X_{n+1}|(X_n = x) \sim f_\theta(\cdot|x), \quad Y_n|(X_n = x) \sim g_\theta(\cdot|x), \tag{2.5.1}$$

where θ is the parameter and $X_1 \sim \mu_\theta(\cdot)$. We shall assume that θ is fixed in this section and it will be omitted in the notations.

For this HMM, one is interested in the posterior density $p(x_{1:T}|y_{1:T}) = \frac{p(x_{1:T},\, y_{1:T})}{p(y_{1:T})} \propto p(x_{1:T},\ y_{1:T})$ where

$$p(x_{1:T},\ y_{1:T}) = \mu_\theta(x_1) \prod_{n=2}^{T} f(x_n|x_{n-1}) \prod_{n=1}^{T} g(y_n|x_n) \tag{2.5.2}$$

and the likelihood function $p(y_{1:T}) := \int p(x_{1:T},\ y_{1:T})\, dx_{1:T}$ at each time T. We shall see that the SMC method will be able to give approximations for these two quantities of interest in a neat manner.

2.5.1 Sequential importance sampling

Sequential importance sampling (SIS) is an extension of importance sampling where the simulation of N samples, called *particles*, is done sequentially, with each new observations. Typically, each particle is of dimension t. Given approximated samples of π_t, one will generate a random variable from a proposal density and append it to the existing particle. Since HMMs are sequential in nature, one can make use of SIS to simulate particles for approximations of the posterior densities involved in HMMs. We shall now illustrate using SIS for the HMM described earlier. As with importance sampling, there will be a proposal density where simulated values are generated with associated weights at each iteration. The proposal density will depend on the existing particle and the new observation. The pseudo code to estimate $p(x_{1:n}, y_{1:n})$ is given in Algorithm 2.

Algorithm 2: Sequential Importance Sampling

begin

Initialisation;

Generate $X_1^k \sim q(\cdot|y_1)$ where $q(\cdot|y_1)$ is the proposal density.

Compute associated weights $w_1^k = \frac{p(X_1^k, y_1)}{q(X_1^k|y_1)} = \mu(X_1^k)\frac{g(y_1|X_1^k)}{q(X_1^k|y_1)}$.

for $n = 2 : T$ **do**

for $k = 1 : N$ **do**

1. *Sampling stage:* Generate $X_n^k \sim q(\cdot|X_{1:n-1}^k, y_n)$ and set $X_{1:n}^k = (X_{1:n-1}^k, X_n^k)$.
2. *Compute weight:* $w_n^k = w_{n-1}^k \times \frac{f(X_n^k|X_{n-1}^k)g(y_n|X_n^k)}{q(X_n^k|X_{1:n-1}^k, y_n)}$

end

end

end

From the pseudo code, one can see that the particles can be easily appended for each iteration. Further, the importance weights can be updated easily by using a recursive relation of the form $w_n^k = w_{n-1}^k u_n^k$ where $u_k^n = \frac{f(X_n^k|X_{n-1}^k)g(y_n|X_n^k)}{q(X_n^k|X_{1:n-1}^k, y_n)}$.

The proposal density plays a pivotal role in the algorithm as it will affect the weights computed at each iteration. Since

$$p(x_n|y_n, x_{n-1}) = \frac{p(x_{n-1:n}, y_n)}{p(y_n, x_{n-1})} \propto f(x_n|x_{n-1})g(y_n|x_n),$$

we should choose $q = p(x_n|y_n, x_{n-1})$ if possible to minimise the variance of the importance weights, otherwise a density that is close to $p(x_n|y_n, x_{n-1})$.

In practice, for simplicity, one can choose $q(x_1|y_1) = f(x_1)$ and $q(x_n|y_n, x_{n-1})$ by $f(x_n|x_{n-1})$ as the expression of the weights will be simplified to $w_n^k = w_{n-1}^k \cdot g(y_n|X_n^k)$. However, as the proposal density for this case is not optimal, one will expect the variance of the importance weights to be relatively high compared to that when an optimal proposal density is used. In the next subsection, we shall introduce the idea

of resampling to reduce the variance of the importance weights when a sub-optimal proposal density is used.

2.5.2 Sequential importance sampling with resampling

One of the drawbacks of using SIS methods is the problem of weight degeneracy. Recall from the earlier subsection that the importance weights satisfy a recursive relation $w_n^k = w_{n-1}^k u_n^k$. Thus

$$w_n^k \propto \exp\left[\sum_{j=1}^{n} \log(u_j^k)\right].$$

Since u_j^k is of the form $p(x_j^k)/q(x_j^k)$, one has

$$w_n^k \propto \exp\left[-\sum_{j=1}^{n} \log\left(\frac{q(x_j^k)}{p(x_j^k)}\right)\right].$$

By the Law of Large Numbers, the right hand side term can be approximated by

$$\exp\left\{-n\mathbb{E}_q\left[\log\left(\frac{q(X)}{p(X)}\right)\right]\right\}.$$

The term $\mathbb{E}_q\left[\log\left(\frac{q(X)}{p(X)}\right)\right]$ is known as the *Kullback-Leibler divergence* between q and p and is known to be non-negative. Consequently, the importance weights will decrease at an exponential rate as n increases. Further, the variance of the importance weights typically grows at an exponential rate in n as well. In another words, the weights are likely to degenerate to 0 as n increases. This makes the associated particles to have insignificant contribution to the estimated density function. If there are too many such ineffective particles in the sample, the approximation of the target density will be inefficient computationally as most computational time is spent on updating the particles with weights that have little contribution to the actual estimation. This results in estimates whose variances increases, usually exponentially, with n. The reader can refer to Cappé *et al.* (2005) for an example on how weight degeneracy will affect the efficiency of an algorithm.

To address the problem of weight degeneracy, one could consider using an optimal proposal density in the sampling stage. Since $\mathbb{E}_q\left[\log\left(\frac{q(X)}{p(X)}\right)\right] = 0$ if $q(x) = p(x)$ almost everywhere, using an optimal proposal density will eliminate the problem of weight degeneracy. Typically, one will not be able to use the optimal proposal density. However, if one is able to obtain a good approximation to the optimal proposal density, one may be able to control the variance of the importance weights to improve the performance of the SIS algorithm.

Another approach is to introduce a resampling stage to the SIS algorithm. Gordon *et al.* (1993) were one of the first to introduce the idea of resampling to address the problem of weight degeneracy. First, consider the importance sampling (IS) estimate $\hat{\pi}_n(x_{1:n})$ to $\pi_n(x_{1:n})$ using $q_n(x_{1:n})$ as the proposal density. Since $X_{1:n}^i$ are weighted samples from q_n, one can use the weights of the samples $X_{1:n}^i$ to obtain approximate samples from $\pi_n(x_{1:n})$. In another words, one could sample from the IS estimate $\hat{\pi}_n(x_{1:n})$ instead, that is, sampling $X_{1:n}^i$ with corresponding standardised weights W_n^i. This is equivalent to sampling from the multinomial distribution $N_n^i \sim (N, W_n^{1:N})$ for $1 \leq i \leq N$ where N_n^i denotes the number of offspring from the particle $X_{1:n}^i$. After resampling, a weight of $1/N$ will be assigned to each offspring. We will approximate $\hat{\pi}_n(x_{1:n})$ by the resampled empirical measure

$$\bar{\pi}_n(x_{1:n}) = \sum_{n=1}^{N} \frac{N_n^i}{N} \delta_{X_{1:n}^i}(x_{1:n}),$$

where $\delta_a(x)$ denotes the Dirac delta mass located at a. Since $\mathbb{E}[N_n^i | W_n^{1:N}] = NW_n^i$, one can see that $\bar{\pi}_n(x_{1:n})$ is unbiased for $\hat{\pi}_n(x_{1:n})$.

Although multinomial sampling is straightforward, there are other resampling schemes that can achieve the unbiased property of $\mathbb{E}[N_n^i | W_n^{1:N}] = NW_n^i$ and yet achieve lower variances for the importance weights compared to the multinomial resampling scheme. Some popular resampling schemes that are used widely in the literature are as follows:

(i) **Systematic Resampling:** Sample $U_1 \sim U[0, \frac{1}{N}]$ and define $U_i = U_1 + \frac{i-1}{N}$ for $i = 2, \ldots, N$. Then set $N_n^i = \left|\{U_j : \sum_{k=1}^{i-1} W_n^k \leq U_j \leq \sum_{k=1}^{i} W_n^k\}\right|$ with the convention $\sum_{k=1}^{0} := 0$.

(ii) **Stratified Resampling:** Sample $U_i \sim U[\frac{i-1}{N}, \frac{i}{N}]$ for $i = 1, \ldots, N$.
Set $N_n^i = \left|\{U_j : \sum_{k=1}^{i-1} W_n^k \leq U_j \leq \sum_{k=1}^{i} W_n^k\}\right|$.

(iii) **Residual Resampling:** Set $\hat{N}_i = \lfloor NW_n^i \rfloor$, sample $\bar{N}_n^{1:N}$ from a multinomial distribution $(N, \bar{W}_n^{1:N})$ where $\bar{W}_n^i \propto W_n^i - N^{-1}\hat{N}_n^i$ and set $N_n^i = \hat{N}_n^i + \bar{N}_n^i$.

(iv) **Multinomial Resampling:** Sample $N_n^{1:N}$ from a multinomial distribution $(N, W_n^{1:N})$.

Systematic resampling was introduced by Carpenter *et al.* (1999). It is the most widely used resampling scheme due to its ease of implementation and the samples can be obtained in $\mathcal{O}(N)$ operations. Further, it outperforms other resampling schemes in most scenarios (Arulampalam *et al.*, 2002).

Kitagawa (1996) introduced the use of stratified resampling for the resampling stage. This samples can be obtained in $\mathcal{O}(N)$ operations with lower variances for the importance weights. The performance of this scheme is comparable to systematic and residual resampling schemes.

Baker (1985, 1987) introduced residual resampling in his work on genetic algorithms. Liu and Chen (1998) were among those who have used residual resampling in their work. This scheme is known to be efficient as the importance weights will have lower variances after resampling is done.

Multinomial resampling is based on the idea of the bootstrap method introduced by Efron (1979). It is simple to implement and the samples could be obtained in $\mathcal{O}(N)$ operations as well. However, in comparison to other resampling schemes, the use of this scheme typically results in higher variances of the importance weights.

A discussion of the performance of these resampling schemes can be found in Douc *et al.* (2005). In this thesis, for simplicity, we shall consider the multinomial resampling scheme in our discussions and simulation studies.

The standard errors can be used to assess the performance of an importance sampling estimate. For the usual SIS algorithm, one is able to establish a Central Limit Theorem (refer to Robert and Casella (2004), Chapter 14) and accordingly, an approximation for the standard error of the SIS estimate. When resampling of particles is done at each iteration, Chan and Lai (2013) provided a consistent estimate for the standard error using the ancestral origin of each particle (refer to Chan and Lai (2013), Theorem 2). Accordingly, we will need to keep track of the ancestral origin of each particle in the execution of the algorithm. We shall denote A_T^i to be the ancestral origin of the i-th particle at time T.

There are two ways to implement resampling: (i) to have a resampling stage at each iteration; (ii) to perform resampling when a criterion is met. The pseudo code for the SIS method with resampling performed at each iteration is given in Algorithm 3. The ancestral origins of the particles are tracked in this algorithm.

The motivation behind resampling is to retain or duplicate particles with high weights and remove particles with low weights. In doing so, one will introduce additional variance to the importance weights. If the variance of the unnormalised importance weights of the particles is small at time t, it may be unnecessary to perform the resampling step. In practice, one will perform resampling when the variance of the importance weights is above a certain value. Kong *et al.* (1994) introduced the use of number of effective particles as a way of approximating the variance of the importance weights. The idea is to compare the variances of the weights when importance sampling is done sequentially against direct importance sampling. An estimate for the number of effective particles, termed as the *effective sample size* (ESS), is given by

$$\hat{N}_{\text{eff}} := \frac{(\sum_{k=1}^{N} w_n^k)^2}{\sum_{k=1}^{N} (w_n^k)^2}.$$

The ESS will take values between 1 (when there is exactly one non-zero weight) to N (when all weights are equal). We will set a threshold number N_t such that resampling will be performed if $\hat{N}_{\text{eff}} < N_t$. We shall term the SIS method with resampling

Algorithm 3: Sequential Importance Sampling with Resampling

begin

Initialisation;

Generate $\tilde{X}_1^k \sim q(\cdot|y_1)$ where $q(\cdot|y_1)$ is the proposal density.

Compute associated weights $\tilde{w}_1^k = \frac{p(\tilde{X}_1^k, y_1)}{q(\tilde{X}_1^k|y_1)} = \mu(\tilde{X}_1^k)\frac{g(y_1|\tilde{X}_1^k)}{q(\tilde{X}_1^k|y_1)}$.

Compute normalised weight $\tilde{W}_1^k = \frac{\tilde{w}_1^k}{\sum_{m=1}^N \tilde{w}_1^m}$.

Generate $B_1^1, \ldots, B_1^N$ such that $P(B_1^k = j) = \tilde{W}_1^j$ and let

$$(X_1^k, A_1^k) = (\tilde{X}_1^{B_1^k}, A_0^{B_1^k}) \text{ for } 1 \leq k \leq N.$$

where $A_0^k = k$.

for $t = 2 : T$ **do**

for $k = 1 : N$ **do**

1. *Sampling stage:* Generate $\tilde{X}_t^k \sim q(\cdot|X_{1:t-1}^k, y_t)$ and set $\tilde{X}_{1:t}^k = (X_{1:t-1}^k, \tilde{X}_t^k)$.
2. *Compute weight:* $\tilde{w}_t^k = \frac{f(\tilde{X}_t^k|X_{t-1}^k)g(y_t|\tilde{X}_t^k)}{q(\tilde{X}_t^k|y_t, X_{t-1}^k)}$
3. *Normalise weight:* $\tilde{W}_t^k = \frac{\tilde{w}_t^k}{\sum_{m=1}^N \tilde{w}_t^m}$.
4. *Resampling stage:* Generate $B_t^1, \ldots, B_t^N$ such that $P(B_t^k = j) = \tilde{W}_t^j$ and let
$$(X_{1:t}^k, A_t^k) = (\tilde{X}_{1:t}^{B_t^k}, A_{t-1}^{B_t^k}) \text{ for } 1 \leq k \leq N.$$

end

end

end

performed when ESS is less than a given threshold as *sequential importance sampling with adaptive resampling*. The pseudo code of the algorithm is given in Algorithm 4. In this algorithm, the ancestral origins of the particles are tracked as well.

In our thesis, henceforth, we will adopt the practice of resampling at each iteration unless otherwise stated. Under this framework, at any time n, one will have two approximations of $p(x_{1:T}|y_{1:T})$ using the SMC particles due to resampling. Referring to algorithm 3, we have

$$\hat{p}_\theta(x_{1:n}|y_{1:n}) = \sum_{k=1}^{N} \tilde{W}_n^k \delta_{\tilde{X}_{1:n}^k}(x_{1:n}) \tag{2.5.3}$$

after the sampling step and

$$\bar{p}_\theta(x_{1:n}|y_{1:n}) = \frac{1}{N} \sum_{k=1}^{N} \delta_{X_{1:n}^k}(x_{1:n}) \tag{2.5.4}$$

after the resampling step. In practice, one would prefer (2.5.3) over (2.5.4) as resampling will introduce additional variance to the weights.

2.5.3 Estimates involving latent states

Recall that for the HMM model with fixed parameter θ and time T, we will use SMC methods to approximate the target density $p(x_{1:T}|y_{1:T}) = \frac{p(x_{1:T},y_{1:T})}{p(y_{1:T})}$ where $p(x_{1:T},\ y_{1:T})$ is given in equation (2.5.2). For ease of notation, we shall denote the marginal likelihood function by

$$\eta_T := \int p(x_{1:T}, y_{1:T})\, dx_{1:T}.$$

For practical scenarios, one would be interested in estimating $\psi_T = \mathbb{E}[\psi(X_{1:T})|Y_{1:T}]$ for some function ψ using the SMC particles. We shall consider the case when η_T is known. An immediate application is to obtain estimates of smoothed means if we set $\psi(X_{1:T}) = X_k$ for $k < T$. We shall motivate the idea in obtaining unbiased estimate for ψ_T using the SMC particles obtained at time T.

Algorithm 4: Sequential Importance Sampling with Adaptive Resampling

begin

Initialisation;

Generate $\tilde{X}_1^k \sim q(\cdot|y_1)$ where $q(\cdot|y_1)$ is the proposal density.

Compute associated weights $\tilde{w}_1^k = \frac{p(\tilde{X}_1^k, y_1)}{q(\tilde{X}_1^k|y_1)} = \mu(\tilde{X}_1^k)\frac{g(y_1|\tilde{X}_1^k)}{q(\tilde{X}_1^k|y_1)}$.

Compute normalised weight $\tilde{W}_1^k = \frac{\tilde{w}_1^k}{\sum_{m=1}^N \tilde{w}_1^m}$.

If $\hat{N}_{\text{eff}} < N_1$,

Generate $B_1^1, \ldots, B_1^N$ such that $P(B_1^k = j) = \tilde{W}_1^j$ and let

$$(X_1^k, A_1^k) = (\tilde{X}_1^{B_t^k}, A_0^{B_t^k}) \text{ for } 1 \leq k \leq N,$$

where $A_0^k = k$. Set $w_n^k = 1$ for all k.

Else $(X_{1:n}^k, A_t^k) = (\tilde{X}_{1:n}^k, A_{t-1}^k)$ and $w_n^k = \tilde{w}_n^k$ for all k.

for $t = 2 : T$ **do**

for $k = 1 : N$ **do**

1. *Sampling stage:* Generate $\tilde{X}_t^k \sim q(\cdot|X_{1:t-1}^k, y_t)$ and set $\tilde{X}_{1:t}^k = (X_{1:t-1}^k, \tilde{X}_t^k)$.

2. *Compute weight:* $\tilde{w}_t^k = w_{t-1}^k \cdot \frac{f(\tilde{X}_t^k|X_{t-1}^k)g(y_t|\tilde{X}_t^k)}{q(\tilde{X}_t^k|y_t, X_{t-1}^k)}$

3. *Normalise weight:* $\tilde{W}_t^k = \frac{\tilde{w}_t^k}{\sum_{m=1}^N \tilde{w}_t^m}$.

4. *Resampling stage:* If $\hat{N}_{\text{eff}} < N_t$,;

Generate $B_t^1, \ldots, B_t^N$ such that $P(B_t^k = j) = \tilde{W}_t^j$ and let

$$(X_{1:t}^k, A_t^k) = (\tilde{X}_{1:t}^{B_t^k}, A_{t-1}^{B_t^k}) \text{ for } 1 \leq k \leq N.$$

Set $w_n^k = 1$ for all k.

Else $(X_{1:n}^k, A_t^k) = (\tilde{X}_{1:n}^k, A_{t-1}^k)$ and $w_n^k = \tilde{w}_n^k$ for all k.

end

end

end

Referring to Algorithm 3, at each iteration, the importance weights are the likelihood ratio between the target density and proposed density. When resampling is done, to ensure that the estimate will be unbiased, one have to reweigh the particles based on the probabilities that they will appear in the resample. Define $\bar{w}_n = \frac{1}{N}\sum_{k=1}^{N} \tilde{w}_n^k$. We have the following theorem for an unbiased estimator of ψ_T, where resampling is done at each iteration for the SMC algorithm.

Theorem 2.2

An unbiased estimate of ψ_T using the generated SMC particles is given by

$$\begin{aligned}\hat{\psi}_T &= \frac{1}{N\eta_T}\sum_{k=1}^{N}\left[\prod_{n=1}^{T}\frac{\bar{w}_n}{\tilde{w}_n^k}\right]\prod_{n=1}^{T}\tilde{w}_n^k\psi(X_{1:T}^k) \\ &= \frac{1}{N\eta_T}\sum_{k=1}^{N}\left[\prod_{n=1}^{T}\bar{w}_n\right]\psi(X_{1:T}^k).\end{aligned}$$

The proof of this theorem, motivated by Chan and Lai (2013), is given in Chapter 3, Theorem 3.4. The proof applies martingale differences.

2.5.4 An unbiased estimate of the likelihood function

A useful by-product in the particle filter is the estimate of the marginal likelihood function $p_\theta(y_{1:T})$. We shall show that the estimate obtain is unbiased. The marginal likelihood estimate is pivotal to the implementation of particle Markov chain Monte Carlo algorithm. This will be discussed in Section 2.8. We shall consider the case when resampling is done at each iteration.

Theorem 2.3

An unbiased estimate of η_T using the generated SMC particles is given by

$$\hat{\eta}_T := \prod_{n=1}^{T}\bar{w}_n.$$

Proof:

From Theorem 2.2, $\hat{\psi}_T = \frac{1}{N\eta_T}\sum_{k=1}^{N}\hat{\eta}_T\psi(X_{1:T}^k)$ is unbiased for ψ_T. Thus, when $\psi \equiv 1$, $\hat{\psi}_T = \frac{\hat{\eta}_T}{\eta_T}$. Since $\hat{\psi}_T$ is unbiased for ψ_T, $\mathbb{E}\left[\frac{\hat{\eta}_T}{\eta_T}\right] = 1$, therefore $\hat{\eta}_T$ is unbiased for η_T.

□

2.6 Markov chain Monte Carlo methods

In the previous sections, we have discussed various sampling techniques that will provide approximation to a target density. We have discussed particle filters as a tool in approximating the posterior density of an HMM in the previous section. In our discussion, we have assumed that θ and T are both fixed. Suppose now we are interested in Bayesian inference of an HMM. That is, we are interested in the posterior density $p(\theta, x_{1:T}|y_{1:T})$ where the HMM parameter θ is given a prior $\pi(\cdot)$. Typically, the posterior density is intractable and one needs to make use of numerical methods to obtain approximation to this density. For this scenario, one wishes to obtain samples of both the parameter θ and the corresponding hidden states $X_{1:T}$ for a fixed T to approximate the posterior density.

Markov chain Monte Carlo (MCMC) methods are popular techniques used in parameter estimation in a Bayesian framework. Recall that SMC methods make use of a proposal density to simulate particles with appropriate weights to approximate the target density. While MCMC methods use a proposal density as well, the principle differs from the SMC methods. For such methods, one will construct a Markov chain such that its stationary distribution is the target posterior density. The key to such methods is to ensure that the Markov chain will converge to the target density. To make use of MCMC methods for Bayesian inferences In the context of HMMs, one strategy is to make use of MCMC algorithm for parameter estimation while using SMC methods to simulate the hidden states.

In this section, we shall first state the conditions for a Markov chain to converge to

its stationary distribution if it exists. We will then introduce the notion of MCMC methods and discuss a couple of important MCMC algorithms that are used in practice. This section will also serve as a basis for subsequent sections, where we will consider different methods for Bayesian inferences.

2.6.1 Convergence of Markov chains

In this subsection, we will consider the necessary conditions for a Markov chain to converge to its stationary distribution. Readers who are interested for an in-depth treatment on this topic can refer to Meyn and Tweedie (2009), Cappé (2005) and Robert and Casella (2004).

Consider a Markov chain $\{X_n\}$ taking values on a measurable space $(\mathbb{X}, \mathcal{X})$ with transition kernel K. The kernel for n transitions is denoted by K^n. We shall list a number of definitions that are essential to establish the convergence of a Markov chain to its stationary distribution.

For any set $A \in \mathcal{X}$, we define the first *hitting time* σ_A and *return time* τ_A respectively by

$$\begin{aligned} \sigma_A &= \inf\{n \geq 0 : \ X_n \in A\}, \\ \tau_A &= \inf\{n \geq 1 : \ X_n \in A\}. \end{aligned}$$

Definition 2.6.1. A Markov chain with transition kernel K is said to be φ-irreducible if there exists a measure φ on $(\mathbb{X}, \mathcal{X})$ such that for any $A \in \mathcal{X}$ with $\varphi(A) > 0$, $\mathbb{P}_x(\tau_A < \infty) > 0$ for all $x \in \mathbb{X}$.

Definition 2.6.2. A set C is *small* if there exist $m \in \mathbb{N}$ and a non-zero measure ν_m such that

$$K^m(x, A) \geq \nu_m(A), \quad \forall x \in C, \forall A \in \mathcal{X}.$$

Definition 2.6.3. A φ-irreducible chain $\{X_n\}$ has a *cycle of length d* if there exist a small set C, an associated integer M, and a probability distribution ν_M such that d is the g.c.d. of

$$\{m \geq 1 : \exists \delta_m > 0 \text{ such that } C \text{ is small for } \nu_M \geq \delta_m \nu_M\}.$$

The *period* of $\{X_n\}$ is the largest integer d satisfying the above condition.

Definition 2.6.4. A Markov chain is *aperiodic* if $d = 1$. If there exist a small set A and a measure ν_1 such that $\nu_1(A) > 0$, the chain is said to be *strongly aperiodic*.

Definition 2.6.5. A σ-finite measure π is *invariant* for the transition kernel K if

$$\pi(B) = \int_{\mathbb{X}} K(x, B)\pi(dx), \quad \forall B \in \mathcal{X}.$$

The invariant measure is also referred to as *stationary* if it is a probability measure.

Definition 2.6.6. A φ-irreducible Markov chain is called *positive* if it admits an invariant probability measure; otherwise it is called *null*.

Another property of a Markov chain that is strongly linked to stationary distribution is reversibility. For such a chain, time does not matter in the dynamics of the chain.

Definition 2.6.7. A stationary Markov chain is *reversible* if the distribution of X_{n+1} conditioned on $X_{n+2} = x$ is the same as the distribution of X_{n+1} conditioned on $X_n = x$.

In fact, reversibility can be linked to the existence of a stationary measure π if the following condition holds.

Definition 2.6.8. A Markov chain with transition kernel K satisfies the *detailed balance condition* if there exists a non-zero function f satisfying

$$K(y, x)f(y) = K(x, y)f(x) \tag{2.6.1}$$

for every (x, y).

The following theorem shows that when the transition kernel of a Markov chain satisfies the detailed balance condition for a density function, then it is the stationary measure. The proof of this theorem can be found in Robert and Casella (2004).

Theorem 2.4

Suppose that a Markov chain with transition kernel K satisfies the detailed balance condition for a probability density function π. Then

(i) the density π is the invariant density of the chain,

(ii) the chain is reversible.

Theorem 2.4 provides a way to check that a MCMC method has a given target distribution as its stationary distribution. We shall now discuss the conditions needed for a Markov chain to converge to its stationary distribution. We will first introduce the definition of an atom for a Markov chain before stating the condition for convergence of a Markov chain.

Definition 2.6.9. A set $\alpha \in \mathcal{X}$ is called an *atom* if there exists a probability measure ν on $(\mathbb{X}, \mathcal{X})$ such that $K(x, A) = \nu(A)$ for all $x \in \alpha$ and $A \in \mathcal{X}$.

A sufficient condition for the Markov chain to converge to its stationary distribution π is *ergodicity*.

Definition 2.6.10. An atom α is *ergodic* if

$$\lim_{n\to\infty} |K^n(\alpha,\alpha) - \pi(\alpha)| = 0.$$

The following theorem gives conditions for a Markov chain to be ergodic.

Theorem 2.5

If $\{X_n\}$ is a φ-irreducible, positive and aperiodic Markov chain, then for almost all $x \in \mathbb{X}$,

$$\lim_{n\to\infty} \|K^n(x,\cdot) - \pi\|_{TV} = 0,$$

where the total variation norm is given by

$$\|\mu_1 - \mu_2\|_{TV} = \sup_A |\mu_1(A) - \mu_2(A)|.$$

The objective of MCMC is to construct an ergodic Markov chain such that the stationary distribution is the target distribution. In the next subsection, we shall consider two MCMC methods widely used in practice.

2.6.2 MCMC methods

Definition 2.6.11. A *Markov Chain Monte Carlo method* for the simulation of a distribution π is any method producing an ergodic Markov Chain X_k whose stationary distribution is π.

In a typical MCMC algorithm, there will be an *update move* where the chain X_n will either remain at its current state or move to a proposed state with probability based on a Markov kernel. To justify that the Markov chain is reversible with stationary distribution π, we have to show that the chain satisfies the detailed balance equation given in equation (2.6.1).

The efficiency of an MCMC algorithm depends on how fast the Markov chain reaches the stationary distribution. This is termed as *mixing* when the Markov chain reached

its stationary distribution. It is typically hard to ascertain whether the Markov chain has reached its stationary distribution. A way to ensure that the Markov chain has rapid mixing is to use a good proposal density. One can make use of a tuning parameter in the proposal density to adjust the probability of accepting a proposed move. For some scenarios (using a Metropolis-Hastings algorithm via a random walk proposal move), one can find an optimal value for the tuning parameter to ensure good mixing of the Markov chain. In practice, a typical procedure is to ignore the first B iterations, known as the *burn-in period*.

We shall consider two MCMC algorithms that are often used in practice: *Metropolis-Hastings algorithm* and *Gibbs Sampling*.

2.6.3 Metropolis-Hastings algorithm

We will first describe the Metropolis-Hastings (MH) algorithm. The pseudo code is given in Algorithm 5.

Algorithm 5: Metropolis-Hastings Algorithm

begin

Initialisation;

Generate X_0 from a initial distribution;

for $k = 1 : N$ **do**

Generate $\tilde{X}_k \sim q(\cdot|X_{k-1})$;

Compute the MH ratio $r_k = 1 \wedge \frac{\pi(\tilde{X}_k)q(X_{k-1}|\tilde{X}_k)}{\pi(X_{k-1})q(\tilde{X}_k|X_{k-1})}$;

Generate $U_k \sim \mathcal{U}(0, 1)$;

MH Update: If $U_k \leq r_k$, set $X_k = \tilde{X}_k$; else set $X_k = X_{k-1}$;

end

end

In this algorithm, we will generate the proposed move from a density $q(\cdot|x)$ known as the *proposal density*. At every update step, we will compute the ratio known as the *Metropolis-Hastings ratio* or the MH ratio for short. Since the acceptance of the

proposed move is based on this ratio, it is also known as the *acceptance ratio*.

To see that this algorithm will generate a Markov chain with π as its stationary distribution, we will show that it satisfies the detailed balance equation given in equation (2.6.1).

Suppose the current state of the Markov chain is x and the proposed update is y. Without loss of generality, we shall assume that $\pi(y)q(x|y) \leq \pi(x)q(y|x)$. Thus

$$\begin{aligned} \pi(x)K(x,y) &= \pi(x)q(y|x) \cdot \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)} \\ &= \pi(y)q(x|y) \\ &= \pi(y)K(y,x) \end{aligned}$$

since the MH ratio $\frac{\pi(x)q(y|x)}{\pi(y)q(x|y)} \geq 1$ and the Markov chain will move from state y to state x with probability 1. Thus by Theorem 2.4, this Markov chain has π as its stationary distribution.

By Theorem 2.5, to ensure that the generated Markov chain converges to the stationary distribution π, the chain has to be π-irreducible and aperiodic. Roberts and Smith (1994) showed that under some weak conditions, the Markov chain constructed by the MH algorithm is aperiodic and π-irreducible.

Theorem 2.6

If q is aperiodic; or $\mathbb{P}(X_t = X_{t-1}) > 0$ for some $t \geq 1$, then the Metropolis-Hastings algorithm is aperiodic. If q is π-irreducible and $q(x,y) = 0$ if and only if $q(y,x) = 0$, then the Metropolis-Hastings algorithm is π-irreducible.

The proof of this theorem can be found in Roberts and Smith (1994), Theorem 3. Accordingly, the proposal density has to be chosen carefully to ensure that the Markov chain is irreducible. We shall remark that aperiodicity is a consequence of a greater than zero probability of rejecting a proposed move.

If the proposal density is symmetric, that is $q(x|y) = q(y|x)$, then the MH ratio simplified to $\pi(y)/\pi(x)$ and the algorithm is also known as the *Metropolis algorithm*. A classical example of the Metropolis algorithm is the random walk Metropolis algorithm where the proposal density is Gaussian. For this algorithm, the update move could be written as

$$\tilde{X}_k = X_{k-1} + \epsilon, \quad \epsilon \sim N(0, \sigma^2).$$

The variance of the proposal density, σ^2, is known as the *tuning parameter*. The efficiency of the random walk Metropolis algorithm depends on this tuning parameter. If σ^2 is small, even though the proposed move will be accepted with a high probability, the convergence of the Markov chain will be slow since the increment is small. On the other hand, if σ^2 is large, a proposed move will be rejected with high probability and the chain will be stuck at a position for a long time. To check on whether the chain has reached its stationary distribution, one can track the long term fraction of accepted moves, known as the *acceptance rate*. Accordingly, one will adjust the tuning parameter to achieve a desired acceptance rate that will give reasonable convergence to the target distribution. Roberts, Gelman and Gilks (1997) showed that under certain regularity conditions for the target density π, the optimal acceptance rate for this random walk Metropolis algorithm is 0.234. This result gives a useful gauge for the acceptance rate of the random walk Metropolis algorithm in practice.

If the proposal density is independent of the current state x, that is $q(y|x) = q(y)$, then the MH ratio simplifies to $\frac{\pi(y)}{\pi(x)} \cdot \frac{q(x)}{q(y)}$ and the algorithm is also known as *independent Metropolis-Hastings (IMH) algorithm*.

Note that the MH ratio can be rewritten as $\frac{\pi(y)}{\pi(x)} \cdot \frac{q(x|y)}{q(y|x)}$. Accordingly, the target density need only be known up to a constant multiplier which is usually the case in practice.

2.6.4 Gibbs sampling

In the previous subsection, we have introduced the Metropolis-Hastings algorithm. This algorithm can be used to approximate a target density if it is known up to a constant multiplier. Accordingly, the MH algorithm can be used in many situations and is regarded as one of the generic MCMC methods. In this subsection, we shall consider a class of MCMC methods that is more specific in nature. This class of algorithms is used to target a joint distribution of the variables of interest and is known as *Gibbs sampling*. The motivation behind this algorithm is to sample from the *full conditional distribution* (which will be defined later) instead of the joint distribution if the former is easier to sample from. We shall remark that Gibbs sampling, in contrast with the MH algorithm, does not have an accept-reject procedure.

To facilitate the discussion, we will refer to distribution and density interchangeably. We shall begin by considering a simple case. For this setup, the target density is the joint density of the variables X and Y, denoted by $f(x, y)$. To illustrate the algorithm, we need to first recall that the marginal densities of X and Y are given by

$$f_X(x) = \int f(x, y)\, dy \quad \text{and} \quad f_Y(y) = \int f(x, y)\, dx$$

respectively. Further, the conditional densities of Y given X and X given Y are

$$f_{Y|X}(y|x) = \frac{f(x, y)}{f_X(x)} \quad \text{and} \quad f_{X|Y}(x|y) = \frac{f(x, y)}{f_Y(y)}$$

respectively. The idea of this method is to construct a Markov chain (X_t, Y_t) by generating the variables X_t and Y_t from the conditional distribution one after another, and using the most recent values for the variables for the update. As we are dealing with two variables, the algorithm is called the *two-stage Gibbs sampler*. The pseudo code is given in Algorithm 6.

For this algorithm, the transition kernel for the constructed Markov chain (X_t, Y_t) is given by

$$K((x, y), (x', y')) = f_{X|Y}(x'|y) f_{Y|X}(y'|x'). \tag{2.6.2}$$

Algorithm 6: Two-stage Gibbs sampler

begin
  Initialisation: $X_0 = x_0$;
  for $t \geq 1$ **do**
    Generate $Y_t \sim f_{Y|X}(\cdot|X_{t-1})$;
    Generate $X_t \sim f_{X|Y}(\cdot|Y_t)$;
  end
end

To show that the Markov chain constructed has the joint distribution as its stationary distribution, we will need the following condition.

Definition 2.6.12. Let $g(y_1, \ldots, y_n)$ be the joint density of $Y_1, \ldots, Y_n$ and g_i denotes the marginal distribution of Y_i. We say that g satisfies the *positivity condition* if $g_i(y_i) > 0$ for every $i = 1, \ldots, n$ implies that $g(y_1, \ldots, y_n) > 0$.

The following theorem gives the conditions for the Markov chain constructed by the two-stage Gibbs sampler to converge to its stationary distribution.

Theorem 2.7

Under the positivity condition, if the transition kernel given in equation (2.6.2) is absolutely continuous with respect to the dominating measure, the chain (X_t, Y_t) is recurrent and ergodic with stationary distribution $f(x, y)$.

The outline for the proof of this theorem can be found in Robert and Casella (2004), Section 9.2. As the kernel for the two-stage Gibbs sampler is absolutely continuous in most scenarios, Theorem 2.7 will ensure that the Markov chain constructed by this algorithm will converge to the joint distribution of X and Y.

A useful feature of the two-stage Gibbs sampler is that one can approximate the

marginal distribution of X and Y using the sequences $\{X_t\}$ and $\{Y_t\}$ respectively. To illustrate this, we shall first consider the following theorem, which shows that for the Markov chain (X_t, Y_t) constructed, the sequences $\{X_t\}_{t\geq 1}$ and $\{Y_t\}_{t\geq 1}$ are Markov chains as well.

Theorem 2.8

Each of the sequences $\{X_t\}_{t\geq 1}$ and $\{Y_t\}_{t\geq 1}$ produced by the two-stage Gibbs sampler is a Markov chain with corresponding stationary distributions f_X and f_Y.

Proof:

We shall consider the sequence $\{X_t\}$ in our proof as the other sequence can be done is a similar way. For the sequence $\{X_t\}$, it has transition density

$$K(x, x') = \int f_{Y|X}(y|x) f_{X|Y}(x'|y)\, dy.$$

This density only depends on the value of X_t and thus a transition kernel. Accordingly, the sequence is a Markov chain. Further,

$$\begin{aligned}
f_X(x') &= \int f(x', y)\, dy \\
&= \int f_{X|Y}(x'|y) f_Y(y)\, dy \\
&= \int f_{X|Y}(x'|y) \int f(x, y)\, dx\, dy \\
&= \int f_{X|Y}(x'|y) \int f_{Y|X}(y|x) f_X(x)\, dx\, dy \\
&= \int \left[\int f_{Y|X}(y|x) f_{X|Y}(x'|y)\, dy \right] f_X(x)\, dx \\
&= \int K(x, x') f_X(x)\, dx.
\end{aligned}$$

Thus, f_X is the stationary distribution of this Markov chain. □

Using Theorem 2.8, if the Markov chain converges to its stationary distribution, the subchains $\{X_t\}_{t\geq 1}$ and $\{Y_t\}_{t\geq 1}$ will converge to their respective marginal distribution. Hence one can use the subchains to approximate the marginal distributions.

We shall now turn our attention to the general case and the corresponding algorithm is called the *Gibbs sampler*. Consider a target density of the form $f(x_1, \ldots, x_n)$ where it is the joint density of $X_1, \ldots, X_n$. We shall first define the notion of a full conditional distribution.

Definition 2.6.13. For the variables $X_1, \ldots, X_n$, the full conditional distribution of X_k given $X_1, \ldots, X_{k-1}, X_{k+1}, \ldots, X_n$ has the following density function

$$f_k(x_k|x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n) := \frac{f(x_1, \ldots, x_n)}{\int f(x_1, \ldots, x_n)\, dx_k}.$$

The objective of the Gibbs sampler is to construct a Markov chain $(\mathbf{X}^t)$, where $\mathbf{X}^t = (X_1^t, \ldots, X_n^t)$, that has the joint distribution of $X_1, \ldots, X_n$ as its stationary distribution. Going along the same vein as in the case of two variables, the procedure is to update coordinate wise using the full conditionals until all the x_k are updated. The outline of the algorithm is given in Algorithm 7.

Algorithm 7: The Gibbs Sampler

begin

Initialisation: Generate $\mathbf{X}^0 = (X_1^0, \ldots, X_n^0)$;

for $t \geq 1$ **do**

Generate $X_1^t \sim f_1(\cdot|X_2^{t-1}, \ldots, X_n^{t-1})$;

Generate $X_2^t \sim f_2(\cdot|X_1^t, X_3^{t-1}, \ldots, X_n^{t-1})$;

$\vdots$

Generate $X_k^t \sim f_k(\cdot|X_1^t, \ldots, X_{k-1}^t, X_{k+1}^{t-1}, \ldots, X_n^{t-1})$;

$\vdots$

Generate $X_n^t \sim f_n(\cdot|X_1^t, \ldots, X_{n-1}^t)$;

end

end

One advantage of the Gibbs sampler is that the sampling of the coordinates X_k is from a univariate distribution. Compared to the joint distribution, the univariate

distribution is typically easier to simulate from. On the note of convergence, one can refer to, for instance, Robert and Casella (2004) for the justification that the chain produced by the Gibbs sampler is a Markov chain with f as the stationary distribution.

We shall end this section by the following theorem, which gives the relationship between Gibbs sampler and the Metropolis-Hastings algorithm.

Theorem 2.9

The Gibbs sampler given in Algorithm 7 is equivalent to the composition of n Metropolis-Hastings algorithms with acceptance probabilities uniformly equal to 1.

The proof of this theorem can be found in Robert and Casella (2004), Theorem 10.13. This theorem shows that the Gibbs sampler can be viewed as a special case of the MH algorithms.

2.7 Pseudo marginal Markov chain Monte Carlo method

Beaumont (2003), Andrieu and Roberts (2009), Beaumont *et al.* (2009) introduced a new MCMC algorithm that utilises pseudo random variables. With the introduction of these auxiliary variables (which can be implemented easily), one will possibly gain statistical and computational efficiency. This algorithm is known as *Pseudo-Marginal Markov Chain Monte Carlo (PsMCMC) methods.*

Recall that for the usual MCMC algorithm targeting the density $\pi(\mathbf{x})$, a new value $\mathbf{X}^*$ is proposed from the current value $\mathbf{x}$ based on a density $q(\mathbf{x}^*|\mathbf{x})$. The proposed value is accepted based on the Metropolis-Hastings (MH) ratio $1 \wedge [\pi(\mathbf{x}^*)q(\mathbf{x}|\mathbf{x}^*)]/[\pi(\mathbf{x})q(\mathbf{x}^*|\mathbf{x})]$. If the proposed value is accepted, it will be the next current value. For this method to work, the target density must be known unbiasedly (up to a multiplication constant).

For some scenarios, the true target distribution $\pi(\cdot)$, or expectation with respect to it, is difficult to evaluate analytically. Suppose that we are able to compute an estimate $\hat{\pi}_{\mathbf{v}}(\cdot)$ by generating an auxiliary random variable $\mathbf{v}$ from some distribution $p_{\text{aux}}(\mathbf{v}|\mathbf{x})$ such that $\mathbb{E}[\hat{\pi}_{\mathbf{V}}(\mathbf{x})] = c\pi(\mathbf{x})$ with $c > 0$. The value of the constant c is inconsequential to the algorithm and we can safely assume that $c = 1$. That is, $\hat{\pi}_{\mathbf{V}}(\mathbf{x})$ is an unbiased estimator of $\pi(\mathbf{x})$. We shall further assume that $\hat{\pi}_{\mathbf{v}} > 0$. The objective of PsMCMC algorithm is to create a Markov chain with stationary distribution

$$\tilde{\pi}(\mathbf{x}, \mathbf{v}) \propto p_{\text{aux}}(\mathbf{v}|\mathbf{x})\hat{\pi}_{\mathbf{v}}(\mathbf{x}).$$

The marginal distribution of this stationary distribution with respect to $\mathbf{x}$ is

$$\int p_{\text{aux}}(\mathbf{v}|\mathbf{x})\hat{\pi}_{\mathbf{v}}(\mathbf{x})\, d\mathbf{v} = \mathbb{E}(\hat{\pi}_{\mathbf{v}}(\mathbf{x})) = \pi(\mathbf{x}),$$

the target density that one needs.

At each iteration, when a new value $\mathbf{X}^*$ is proposed using the Metropolis-Hastings algorithm, a new auxiliary variable $\mathbf{V}^*$ will be proposed using the density $q(\mathbf{v}^*|\mathbf{x}^*)$. The pair $(\mathbf{x}^*, \mathbf{v}^*)$ will be jointly accepted or rejected based on the MH acceptance probability given by

$$1 \wedge \frac{\hat{\pi}_{\mathbf{v}^*}(\mathbf{x}^*)q(\mathbf{x}^*, \mathbf{x})}{\hat{\pi}_{\mathbf{v}}(\mathbf{x})q(\mathbf{x}, \mathbf{x}^*)}.$$

The convergence properties of the PsMCMC method can be found in Andrieu and Roberts (2009). Using this method, one will be able to substitute the unknown target density using an unbiased estimator and obtain the targeted stationary distribution of $\mathbf{x}$.

In the context of particle filters, $\mathbf{V}$ are the random variables obtained from a particle filter obtained using T observed values with N particles. The likelihood function can be estimated using these particles with the resulting estimate being unbiased as shown in Section 2.5. This method, termed as *particle Markov chain Monte Carlo method* will be discussed in the next section.

2.8 Particle Markov chain Monte Carlo method

In this section, we will consider algorithms targeting the distribution of $p(\theta, x_{1:T}, y_{1:T})$ for a hidden Markov model (2.5.1) where

$$p(\theta,\ x_{1:T},\ y_{1:T}) = p(\theta)p_\theta(x_{1:T},\ y_{1:T}). \tag{2.8.1}$$

and $p_\theta(x_{1:T},\ y_{1:T})$ is given in equation (2.5.2). Under this setup, θ is unknown and assigned a prior $\pi(\theta)$. For Bayesian inferences for this hidden Markov model, we are interested in the posterior density

$$p(\theta,\ x_{1:T}|y_{1:T}) \propto p(\theta)p_\theta(x_{1:T}|y_{1:T}). \tag{2.8.2}$$

In typical MCMC methods targeting this distribution, the algorithm consists of updating the state components $x_{1:T}$ conditional on θ and θ conditional on $x_{1:T}$ alternatively. This algorithm requires sampling from $p(\theta|y_{1:T}, x_{1:T})$, which may be feasible, and $p_\theta(x_{1:T}|y_{1:T})$ where exact sampling is only possible for linear Gaussian models and finite state space hidden Markov models. For other scenarios, the design of proposal densities is necessary as part of a MH requirement. Andrieu *et al.* (2010) proposed a novel MCMC algorithm, named as particle Markov Chain Monte Carlo (PMCMC) method to address this problem. For the PMCMC method, the sampling from $p_\theta(x_{1:T}|y_{1:T})$ is done by running an SMC algorithm to obtain an approximation to this density, followed by sampling from this approximation. N particles will be generated from this approximation for the Metropolis-Hastings (MH) update. In doing so, the approximation is of low dimension. As mentioned in the previous section, PMCMC algorithm is an example of a PsMCMC algorithm.

2.8.1 Particle independent Metropolis-Hastings sampler

In MCMC methods, we must ensure that the target distribution is the stationary distribution of the Markov chain generated in the algorithm. In a standard independent Metropolis-Hastings (IMH) update, to ensure that $p_\theta(x_{1:T}|y_{1:T})$ is invariant, we have to choose a proposal density $q_\theta(x_{1:T}|y_{1:T})$ to sample candidates $X^*_{1:T}$, which given a

current state $X_{1:T}$, are accepted with probability

$$1 \wedge \frac{p_\theta(X^*_{1:T}|y_{1:T})q_\theta(X_{1:T}|y_{1:T})}{p_\theta(X_{1:T}|y_{1:T})q_\theta(X^*_{1:T}|y_{1:T})}.$$

The optimal choice of $q_\theta(x_{1:T}|y_{1:T})$ is $p_\theta(x_{1:T}|y_{1:T})$. However, this choice is infeasible in practice as the exact form is unavailable. To circumvent this problem, an SMC algorithm is run to obtain an approximation $\hat{p}_\theta(x_{1:T}|y_{1:T})$ as given in equation (2.5.3). Sampling from this empirical distribution is easy through the multinomial distribution from $\{X^k_{1:T}; W^k_T; k = 1, \ldots, N\}$. However, computing the MH ratio for this proposed density requires the marginal distribution of $X^*_{1:T}$ which is hard to compute. In fact, this distribution is given by

$$q_\theta(dx_{1:T}|y_{1:T}) = \mathbb{E}\{\hat{p}_\theta(dx_{1:T}|y_{1:T})\},$$

where the expectation is taken with respect to all the random variables generated by the SMC algorithm. Accordingly, the expression for $q_\theta(dx_{1:T}|y_{1:T})$ does not have a closed form. To overcome this problem, we can employ a standard 'auxiliary variables trick' by embedding the sampling from $p_\theta(x_{1:T}|y_{1:T})$ into that of sampling from an appropriate distribution defined on an extended space including all the random variables involved in the calculation of the above expectation. It turns out that the resulting particle independent Metropolis-Hastings (PIMH) sampler admits a simple form with $\hat{p}_\theta(y_{1:T})$ as given in Theorem 2.3. Since the likelihood estimate is unbiased, from Section 2.7, the algorithm will produce approximate samples from the targeted posterior distribution. The algorithm is outlined in Algorithm 8.

The estimates for the likelihood functions are unbiased and this will ensure that the Markov chain produced will admit the target distribution as its stationary distribution.

2.8.2 Particle marginal Metropolis-Hastings sampler

When θ is unknown, we will be interested in sampling from $p(\theta, x_{1:T}|y_{1:T})$ as given in (2.8.2). The approach is to jointly update θ and $x_{1:T}$ at each iteration. Recall the

Algorithm 8: Particle Independent MH Algorithm

begin

Step 1: *Initialisation*

Set $i = 0$ and run an SMC algorithm targeting $p_\theta(x_{1:T}|y_{1:T})$ to obtain $\hat{p}_\theta(\cdot|y_{1:T})$.

Sample $X_{1:T}(0) \sim \hat{p}_\theta(\cdot|y_{1:T})$ and let $\hat{p}_\theta(y_{1:T})(0)$ be the corresponding marginal likelihood estimate.

Step 2: *Recursive Steps*

For $i \geq 1$,

1. run a SMC algorithm targeting $\hat{p}_\theta(x_{1:T}|y_{1:T})$ to obtain $\hat{p}_\theta(\cdot|y_{1:T})$ and sample $X^*_{1:T} \sim \hat{p}_\theta(\cdot|y_{1:T})$ and let $\hat{p}_\theta(y_{1:T})^*$ be the corresponding marginal likelihood estimate.

2. compute the MH ratio given by

$$1 \wedge \frac{\hat{p}_\theta(y_{1:T})^*}{\hat{p}_\theta(y_{1:T})(i-1)},$$

Set $X_{1:T}(i) = X^*_{1:T}$ and $\hat{p}_\theta(y_{1:T})(i) = \hat{p}_\theta(y_{1:T})^*$ with probability given by the MH ratio;

else set $X_{1:T}(i) = X_{1:T}(i-1)$ and $\hat{p}_\theta(y_{1:T})(i) = \hat{p}_\theta(y_{1:T})(i-1)$.

end

decomposition

$$p(\theta, x_{1:T}|y_{1:T}) = p(\theta|y_{1:T})p_\theta(x_{1:T}|y_{1:T})$$

and assume that sampling from the conditional density $p_\theta(x_{1:T}|y_{1:T})$ is feasible. In such scenario, a natural candidate for the proposal density for the MH update will take the form

$$q\{(\theta^*, x^*_{1:T})|(\theta, x_{1:T})\} = q(\theta^*|\theta)p_{\theta^*}(x^*_{1:T}|y_{1:T}),$$

where $x^*_{1:T}$ is dependent on the proposed θ^*. Thus $q(\theta^*|\theta)$ will determine performance of the algorithm. The resulting MH ratio is given by

$$\frac{p(\theta^*, x^*_{1:T}|y_{1:T})}{p(\theta, x_{1:T}|y_{1:T})} \frac{q\{(\theta, x_{1:T})|(\theta^*, x^*_{1:T})\}}{q\{(\theta^*, x^*_{1:T})|(\theta, x_{1:T})\}} = \frac{p(\theta^*)p_{\theta^*}(y_{1:T})}{p(\theta)p_\theta(y_{1:T})} \frac{q(\theta|\theta^*)}{q(\theta^*|\theta)}. \tag{2.8.3}$$

This ratio suggests the use of particle marginal Metropolis-Hastings (PMMH) update where the SMC approximations for the marginal likelihood $p_\theta(y_{1:T})$ is used for sampling from $p_\theta(x_{1:T}|y_{1:T})$. The PMMH sampler is as outlined in Algorithm 9.

Algorithm 9: Particle Marginal MH Algorithm

begin

Step 1: *Initialisation*

Set $i = 0$ and $\theta(0)$ arbitrarily.

Run a SMC algorithm targeting $p_{\theta(0)}(x_{1:T}|y_{1:T})$ and sample

$X_{1:T}(0) \sim \hat{p}_{\theta(0)}(\cdot|y_{1:T})$.

Compute the marginal likelihood estimate $\hat{p}_{\theta(0)}(y_{1:T})$.

Step 2: *Recursive Step*

For $i \geq 1$,

1. sample $\theta^* \sim q\{\cdot|\theta(i-1)\}$,
2. run a SMC algorithm targeting $p_{\theta^*}(x_{1:T}|y_{1:T})$ and sample $X^*_{1:T} \sim \hat{p}_{\theta^*}(\cdot|y_{1:T})$ and compute $\hat{p}_{\theta^*}(y_{1:T})$, the marginal likelihood estimate,
3. compute the MH ratio given by

$$1 \wedge \frac{\hat{p}_{\theta^*}(y_{1:T})p(\theta^*)}{\hat{p}_{\theta(i-1)}(y_{1:T})p\{\theta(i-1)\}} \frac{q\{\theta(i-1)|\theta^*\}}{q\{\theta^*|\theta(i-1)\}},$$

and set $\theta(i) = \theta^*$, $X_{1:T}(i) = X^*_{1:T}$ and $\hat{p}_{\theta(i)}(y_{1:T}) = \hat{p}_{\theta^*}(y_{1:T})$ with probability given by the MH ratio; else set $\theta(i) = \theta(i-1)$, $X_{1:T}(i) = X_{1:T}(i-1)$ and $\hat{p}_{\theta(i)}(y_{1:T}) = \hat{p}_{\theta(i-1)}(y_{1:T})$.

end

Since the unbiased estimates for the likelihood function are used in the computation

of the MH ratio, the algorithm will produce a Markov chain targeting the desired posterior distribution for θ.

2.9 SMC^2 algorithm

Chopin *et al.* (2013) introduced an efficient algorithm for targeting the posterior density $p(\theta, x_{1:T}|y_{1:T})$ of a hidden Markov model where T is growing. This algorithm is termed as the SMC^2 algorithm. It is an extension of the particle Markov chain Monte Carlo method introduced in Section 2.8. Once again, consider the HMM with parameters $\theta \in \Theta$, prior $p(\theta)$ and hidden Markov state process govern by $p(x_1|\theta) = \mu_\theta(x_1)$, $p_(x_{t+1}|x_{1:t}, \theta) = f_\theta(x_{t+1}|x_t)$ for $t \geq 1$ and the observed process y_t such that $p(y_t|y_{1:t-1}, x_{1:t}, \theta) = g_\theta(y_t|x_t)$ for $t \geq 1$ and the distribution of interest is the posterior density $\pi_t(\theta, x_{1:t}) = p(\theta, x_{1:t}|y_{1:t})$ for $t \geq 1$ and the likelihood function $p(y_{1:t})$.

For the SMC^2 algorithm, at each iteration, N_θ θ-particles will be generated and to each θ particle, a particle filter (PF) will be attached to it comprising of N_x x-particles. The nested filters give rise to the superscript in the name of this algorithm. Recall that the PF will provide unbiased estimates for the likelihood function. Accordingly, the θ particles will be properly weighed in the spirit of Section 2.7. The algorithm is given in Algorithm 10.

The PMCMC kernel K_t in the algorithm is described as follows. For the variable set $(\theta, x_{1:t}^{1:N_x})$ with likelihood estimate $\hat{Z}_t(\theta, x_{1:t}^{1:N_X})$,

(i) Sample $\tilde{\theta}$ from the proposal kernel $q(\cdot|\theta)$.

(ii) Run a new PF for $\tilde{\theta}$ to obtain $\tilde{x}_{1:t}^{1:N_x}$ and compute the likelihood estimate $\hat{Z}_t(\tilde{\theta}, \tilde{x}_{1:t}^{1:N_X})$

(iii) Accept the proposed move with probability

$$1 \wedge \frac{p(\tilde{\theta})\hat{Z}_t(\tilde{\theta}, \tilde{x}_{1:t}^{1:N_X})q(\theta|\tilde{\theta})}{p(\theta)\hat{Z}_t(\theta, x_{1:t}^{1:N_X})q(\tilde{\theta}|\theta)}.$$

Algorithm 10: SMC^2 Algorithm

begin

Initialisation: Generate θ^m from $p(\theta)$ for $1 \leq m \leq N_\theta$ and set $w^m = 1$;

for $t = 1 : T$ **do**

for $m = 1 : N_\theta$ **do**

Perform iteration t of the PF described in Section 2.5;

If $t = 1$, sample $x_1^{1:N_x,m}$ and compute

$$\hat{p}(y_1|\theta^m) = \frac{1}{N_x}\sum_{n=1}^{N_x} w_{1,\theta}(x_1^{n,m});$$

If $t > 1$, sample $\tilde{x}_t^{1:N_x,m}$ conditional on $x_{1:t-1}^{1:N_x,m}$ and perform resampling. Compute

$$\hat{p}(y_t|y_{1:t-1},\theta^m) = \frac{1}{N_x}\sum_{n=1}^{N_x} w_{t,\theta}(x_{1:t}^{n,m});$$

Update the importance weights of the θ particles using

$$w^m\hat{p}(y_t|y_{1:t-1},\theta^m) \rightarrow w^m;$$

end

Compute ESS $= \left(\sum_{m=1}^{N_\theta} w^m\right)^2 / \sum_{m=1}^{N_\theta}(w^m)^2$;

If ESS $< \gamma N_\theta$ for pre-determined $\gamma \in (0,\, 1)$;

Sample $(\tilde{\theta}^m, \tilde{x}_{1:t}^{1:N_x,m})$ from

$$\frac{1}{\sum_{m=1}^{N_\theta} w^m}\sum_{m=1}^{N_\theta} w^m K_t[(\theta^m, x_{1:t}^{1:N_x,m}), \cdot]$$

where K_t is a PMCMC kernel;

Update the θ particles, states and weights by

$$(\tilde{\theta}^m, \tilde{x}_{1:t}^{1:N_x,m}, 1) \rightarrow (\theta^m, x_{1:t}^{1:N_x,m}, w^m)$$

end

end

Let the joint probability density of all the random variables generated for the algorithm up to time t be denoted by $\psi_{t,\theta}(x_{1:t}^{1:N_x})$ for $t \geq 1$. By Theorem 2.3, the likelihood estimate

$$\hat{Z}_t(\theta, x_{1:t}^{1:N_x}) = \frac{1}{N_x^t} \prod_{s=1}^{t} \left\{ \sum_{n=1}^{N_x} w_{s,\theta}(x_{1:s}^n) \right\}$$

is an unbiased estimator of $p(y_{1:t}|\theta)$. Further, the expectation of $\hat{Z}_t(\theta, x_{1:t}^{1:N_x})$ with respect to $\psi_{t,\theta}$ is $p(y_{1:t}|\theta)$ as well. As a result of the use of $\hat{Z}_t(\theta, x_{1:t}^{1:N_x})$ in the SMC^2 algorithm, the target density π_t for $t \geq 1$ of the algorithm is given by

$$\pi_t(\theta, x_{1:t}^{1:N_x}) = p(\theta)\psi_{t,\theta}(x_{1:t}^{1:N_x}) \frac{\hat{Z}(\theta, x_{1:t}^{1:N_x})}{p(y_{1:t})}.$$

Since $\hat{Z}_t(\theta, x_{1:t}^{1:N_x})$ is unbiased for $p(y_{1:t}|\theta)$, one can easily deduce the normalising constant $p(y_{1:t})$. It was shown in Chopin *et al.* (2013) that the sequence of auxiliary distributions π_t generated by the SMC^2 algorithm are of increasing dimensions whose marginals include the posterior density of interest for the HMM. This establishes the working principle of the SMC^2 algorithm.

As can be seen from the algorithm, the SMC^2 algorithm is a sequential but not online algorithm. The computational time increases with iterations due to the cost of the MCMC steps which involve generating new PFs for computation of the weights. As stated in Chopin *et al.* (2013), these MCMC steps will occur at a decreasing rate. An attractive feature of the SMC^2 algorithm is that it does not require a specific structure for the HMM and thus can be applied in a wide array of situations.

2.10 Substitution algorithm

From the earlier section, the PMCMC sampler when applied to a HMM with unknown parameter results in a complicated algorithm. The evaluation of the likelihood based on a proposed θ^* takes up substantial computation time when the time index T is large (since a full SMC algorithm has to be run). Accordingly, if a proposed move is rejected, the computation of the MH ratio will be costly. Chan and Lai (2014)

proposed a new MCMC method to handle this scenario. The advantages of the new approach, hereof defined as the *substitution algorithm*, is that no resampling is needed. The empirical particle distribution is used to generate a proposed particle that will replace one of the existing particle. This procedure will retain good particles instead of resampling the particles in the MCMC step.

This algorithm will approximate a target distribution of θ by a set of n representative atoms. This is done sequentially such that the associated weights of the atoms converges weakly to the target distribution as the number of iterations $K \to \infty$.

At each iterative step, we will generate a new atom based on the existing n atoms. This new atom will *substitute* one of the existing atoms based on a MH-type procedure. We will justify that this is a MH-type procedure after the outline of the algorithm.

2.10.1 Algorithm

The outline of the algorithm is as follows.

Initialisation: Generate $\theta_1^0, \ldots, \theta_n^0$ and set $S_0 = \{\theta_1^0, \ldots, \theta_n^0\}$.

Iterative Steps: For $k = 1, \ldots, K$,

- Sample $\theta_{n+1}^{k-1} \sim q(\cdot|S_{k-1})$.
- Let $S_{k-1,i} = \{\theta_1^{k-1}, \ldots, \theta_{n+1}^{k-1}\} \setminus \{\theta_i^{k-1}\}$ for $1 \leq i \leq n+1$.
  Compute $\lambda_{k,i} = \frac{q(\theta_i^{k-1}|S_{k-1,i})}{p(\theta_i^{k-1})}$ for $1 \leq i \leq n+1$.
- Compute the weights $w_i^k = \frac{\lambda_{k,i}}{\sum_{j=1}^{n+1} \lambda_{k,j}}$ for $1 \leq i \leq n+1$.

- Generate J_k using a multinomial distribution on $\{1, \ldots, n+1\}$ with associated probabilities $\{w_1^k, \ldots, w_{n+1}^k\}$.

- If $J_k = n+1$, $\theta_i^k = \theta_i^{k-1}$ for all $k = 1, \ldots, K$. Else $\theta_i^k = \begin{cases} \theta_i^{k-1}, & \text{for } i \neq J_k, \\ \theta_{n+1}^{k-1}, & \text{for } i = J_k. \end{cases}$

  Set $S_k = \{\theta_1^k, \ldots, \theta_n^k\}$.

The estimate of $\psi^* = \mathbb{E}_p \psi(\theta)$ is given by

$$\hat{\psi}_K = \frac{1}{n} \sum_{i=1}^{n} \hat{\psi}_K^i \text{ where } \hat{\psi}_K^i = \frac{\sum_{k=1}^{K} \tilde{\lambda}_k^{-1} \psi(\theta_i^k)}{\sum_{k=1}^{K} \tilde{\lambda}^{-1}}, \text{ and } \tilde{\lambda}_k = \left(\frac{1}{n} \sum_{i=1}^{n} \lambda_{k,i} \right) \vee \lambda_* \quad (2.10.1)$$

where λ_* is a positive number to ensure that the weights $\tilde{\lambda}_k^{-1}$ are not too large.

A simpler estimate of ψ^* is given by

$$\hat{\psi}_{K,B} = \frac{1}{n(K-B)} \sum_{i=1}^{n} \sum_{k=B+1}^{K} \psi(\theta_i^k) \quad (2.10.2)$$

where B is the burn-in period.

To justify that the proposed move in the iterative step is a MH-type procedure, we will show that the detailed balance condition is satisfied for the proposed move. Note that in the update move, there is a total of $n+1$ proposed moves, which is different from the usual MH move where only 2 states are involved.

In the substitution method, the proposed move is from S_{k-1} to $S_{k-1,i}$ where the ith atom is replaced by the generated $(n+1)$-th atom. Assume that the n atoms of θ_i^{k-1}

are independent. Consider the transition kernel K for this Markov chain. We have

$$\begin{aligned}
\mathbb{P}(S_{k-1})K(S_{k-1}, S_{k-1,i}) &= \mathbb{P}(S_{k-1}) \cdot q(\theta_{n+1}^{k-1}|S_{k-1}) \cdot \mathbb{P}(J_k = i) \\
&= \left(\prod_{i=1}^{n} p(\theta_i^{k-1})\right) \cdot q(\theta_{n+1}^{k-1}|S_{k-1}) \cdot \frac{\lambda_{k,i}}{\sum_{j=1}^{n+1} \lambda_{k,j}} \\
&= \left(\prod_{i=1}^{n} p(\theta_i^{k-1})\right) \cdot q(\theta_{n+1}^{k-1}|S_{k-1}) \cdot \frac{q(\theta_i^{k-1}|S_{k-1,i})/p(\theta_i^{k-1})}{\sum_{j=1}^{n+1} \lambda_{k,j}} \\
&= \left(\prod_{j=1, j\neq i}^{n+1} p(\theta_j^{k-1})\right) \cdot q(\theta_i^{k-1}|S_{k-1,i}) \cdot \frac{q(\theta_{n+1}^{k-1}|S_{k-1})/p(\theta_{n+1}^{k-1})}{\sum_{j=1}^{n+1} \lambda_{k,j}} \\
&= \left(\prod_{j=1, j\neq i}^{n+1} p(\theta_j^{k-1})\right) \cdot q(\theta_i^{k-1}|S_{k-1,i}) \cdot \frac{\lambda_{k,n+1}}{\sum_{j=1}^{n+1} \lambda_{k,j}} \\
&= \mathbb{P}(S_{k-1,i}) \cdot q(\theta_i^{k-1}|S_{k-1,i}) \cdot \mathbb{P}(I_k = n+1) \\
&= \mathbb{P}(S_{k-1,i})K(S_{k-1,i}, S_{k-1})
\end{aligned}$$

where I_k is generated using a multinomial distribution on $\{1, \ldots, n+1\}$ with associated probabilities $\{w_1^k, \ldots, w_{n+1}^k\}$ and noting that $q(\theta_{n+1}^{k-1}|S_{k-1}) = q(\theta_{n+1}^{k-1}|S_{k-1,n+1})$.

2.10.2 Application to HMMs

An immediate application of the substitution method is to perform parameter inference for a Hidden Markov model. Consider the HMM described in Definition 2.1.1. As in the PMCMC method, we will have to update the hidden states and the parameter given the observed values, that is, estimate $(\theta, x_{1:T})$ given $y_{1:T}$. As the observations are given sequentially, we will perform a state update and parameter update sequentially, using a particle filter method for the state update and the substitution method for the parameter update. We will generate n samples for θ and for each θ sample, we will generate M particles for the particle filter. In this scenario, each atom will consist of a sample of θ and M particles approximating the states, that is, $A_i^t = \{\theta_i^t, \mathcal{X}_{i,1}^t, \ldots \mathcal{X}_{i,M}^t\}$ for $1 \leq i \leq n$ where $\mathcal{X}_{i,m}^t = \{X_{i,m}^1, \ldots, X_{i,m}^t\}$ is the m-th particle. The outline of the substitution method is given below.

Initialisation:

(i) Sample $\theta_1^0, \ldots, \theta_n^0$ from $p(\cdot)$ and set $S_0 = \{\theta_1^0, \ldots, \theta_n^0\}$.

(ii) Let $\hat{p}_0(\theta) = p(\theta)$, $w_{i,m}^0 = 1$ for $1 \leq i \leq n$ and $1 \leq m \leq M$.

Particle filter state updates at time $t \in \{1, \ldots, T\}$:

Suppose the current set of parameter values are $S_{k-1} = \{\theta_1^{k-1}, \ldots, \theta_n^{k-1}\}$. For $i = 1, \ldots, n$,

(i) For $m = 1, \ldots M$,

(a) Sample $\tilde{X}_{i,m}^t \sim f_{\theta_i^{k-1}}(\cdot|X_{i,m}^{t-1})$ and let $\tilde{\mathcal{X}}_{i,m}^t = \{\mathcal{X}_{i,m}^{t-1}, \tilde{X}_{i,m}^t\}$

(b) Update the weights $\tilde{w}_{i,m}^t = w_{i,m}^{t-1} \cdot g_{\theta_i^{k-1}}(Y_t|\tilde{X}_{i,m}^t)$.

(c) Compute $ESS_i^t = \frac{\left(\sum_{m=1}^M \tilde{w}_{i,m}^t\right)^2}{\sum_{m=1}^M (\tilde{w}_{i,m}^t)^2}$.

(d) If $ESS_i^t \geq c$, set $\mathcal{X}_{i,m}^t = \tilde{\mathcal{X}}_{i,m}^t$ and $w_{i,m}^t = \tilde{w}_{i,m}^t$.

(e) Elseif $ESS_i^t < c$, perform resampling from $\{\tilde{\mathcal{X}}_{i,m}^t\}_{m=1}^M$ with associated weights $\{\tilde{w}_{i,m}^t\}_{m=1}^M$ to obtain $\mathcal{X}_{i,m}^t$ for $1 \leq m \leq M$.

(ii) Update the target density $\hat{p}_t(\theta_i^{k-1}) = \hat{p}_{\tau_i^k}(\theta_i^{k-1}) \left(\frac{1}{M} \sum_{m=1}^M \tilde{w}_{i,m}^t\right)$ where τ_i^k is the most recent resampling time before time t for atom θ_i^{k-1} and $\tau_i^t = 0$ if no resampling has occurred before time t.

(iii) $\hat{p}_t$ gives an approximation to the posterior distribution of θ given the observations up to time t, up to a normalising constant, in the MCMC scheme with sequential substitution.

Parameter update at time t:

For notational simplicity, we will suppress the superscript t and denote the n atoms before the parameter updates by $(A_1^0, \ldots A_n^0)$ and the corresponding parameters by $(\theta_1^0, \ldots \theta_n^0)$. For $k = 1, \ldots, K$,

(i) Generate a new atom θ_{n+1}^{k-1} from $q(\cdot|\bar{\gamma}_k)$ where $\bar{\gamma}_k = \frac{1}{n} \sum_{i=1}^n \gamma(\theta_i^{k-1})$.

(ii) With this value of θ_{n+1}^{k-1}, perform particle filter state update steps to generate $A_{n+1}^{k-1} = (\theta_{n+1}^{k-1}, \mathcal{X}_{n+1,k-1,1}^{t}, \ldots, \mathcal{X}_{n+1,k-1,M}^{t})$.

(iii) Let the target density $\hat{p}_t(\theta_{n+1}^{k-1})$ be the computed value in step 2 of the state update for the atom A_{n+1}^{k-1}.

(iv) Substitute θ_{n+1}^{k-1} with its associated particles into some θ_j^{k-1} using the procedure described in Section 2.10.1. Let $(A_1^k, \ldots, A_n^k)$ be the n atoms after the substitution.

For ease of computation, given observed states $y_{1:T}$, we will perform the hidden states update up to time T for all n atoms; thereafter, we will perform the parameter update at time T. Accordingly, for each generation of the $(n+1)$-th atom, there will be a total of T states associated with this generated θ value.

For our thesis, we will make use of PMCMC algorithms to approximate the posterior distribution of an HMM with unknown parameters. In Chapter 5, we will use the PMCMC algorithm to approximate the posterior distribution of a stochastic volatility model using real data.

Chapter 3

Parallel Particle Filters

In our review of sequential Monte Carlo (SMC) methods in Chapter 2, we have introduced the idea of resampling to overcome the problem of weight degeneracy for these methods. It was pointed out that if an optimal proposal density is chosen, one can reduce the number of resampling steps in the algorithm. However, when a sub-optimal proposal is used, one would need to use resampling to tackle the issue of weight degeneracy.

With the recent advancements in computer hardware, parallel computing has gained significant attention in reducing computational cost of numerical methods. To reduce the computational cost of these methods, one can make use of parallel computing for certain sections of the algorithm that can be run in parallel. On this note, SMC methods can be enhanced by parallel computing as well. Lee *et al.* (2014) have made use of parallel computing for the sampling step of SMC methods. At each iteration, the sampling step can be distributed to J processors to generate the N random variables. Accordingly, each processor will generate N/J random variables (assuming that $N/J \in \mathbb{N}$), therefore saving computational cost. For another example, Vergé *et al.* (2013) introduced the *island particle model* to facilitate the use of parallel computing. Since the resampling step is a major requirement for the method to be efficient, one would like to make use of parallel computing for this step as well. However, as the resampling step requires sampling from all the n particles, the usage of parallel

computing for this step proves to be a challenge. As a result, a number of researchers focus their attentions in reducing the degree of interaction in the SMC algorithms. Some examples of these works can be found in Lee and Whiteley (2014), Lindsten *et al.* (2014) and Whiteley *et al.* (2013).

The method that we are proposing is different in nature from the above-mentioned methods. To facilitate the use of parallel computing for an SMC method, we will consider data segmentation of the observed sequence into smaller portions. A particle filter will then be run for each portion, allowing the use of parallel computation for these particle filters. Therefore, our proposed method is complementary to the above-mentioned methods and can be incorporated to these methodologies easily.

Apart from facilitating the use of parallel computation, our proposed method could address another problem of SMC methods: *path degeneracy*. For a long observation sequence, due to resampling, one would expect to have less distinct particles for the latent states estimation at an earlier time after the algorithm is completed. As a result, the variance of the latent states estimate will be considerably high. As our proposed method make use of data segmentation of the observation sequence, we could overcome the problem of path degeneracy and thus able to obtain an estimate for the smoothed mean $\mathbb{E}[X_t|Y_{1:T}]$, where $t < T$, with possibly lower variance.

In this chapter, we shall introduce formal notations and establish the framework for this parallel sequential Monte Carlo (SMC) method set-up in Section 3.1. We will term this algorithm as the parallel particle filter (PPF). The pseudo code of the proposed method will be given in this section as well. For this set-up, we will propose an estimate for $\mathbb{E}_p[\psi(X_{1:MT})|Y_{1:MT}]$ and the likelihood $\eta(\theta)$ which will be defined in Section 3.2. The main objective of this chapter is to prove that the proposed estimate for $\mathbb{E}_p[\psi(X_{1:MT})|Y_{1:MT}]$ is unbiased when the likelihood is known. As a consequence, one will be able to show that the estimate for the marginal likelihood $\hat{\eta}(\theta)$ will be unbiased. We will prove this using a martingale difference approach, using two different martingale difference expressions. In Section 3.3, we shall prove

that the proposed estimate is unbiased using a martingale difference expression that is useful for proving a Central Limit Theorem. Another martingale difference expression involving ancestral origins of the particles will be introduced in Section 3.4 which is useful in providing an approximation for the standard error of the estimates. For estimation of $\mathbb{E}_p[\psi(X_{1:MT})|Y_{1:MT}]$ when the likelihood is unknown, one can replace the likelihood function by the proposed unbiased estimate $\hat{\eta}(\theta)$. The estimate will be given in Section 3.2. We shall remark that this estimate, though biased, is consistent. A brief discussion on the computational time of the algorithm will be done in Section 3.5. We will end this chapter by discussing on the choice of the proposal density in Section 3.6.

3.1 Notations and framework

Recall that for a HMM, the hidden states, X_n, and observed states Y_n, satisfies the following relations:

$$X_n \sim f_\theta(\cdot|x_{n-1}), \quad Y_n \sim g_\theta(\cdot|x_n), \tag{3.1.1}$$

where θ is the parameter of the HMM and we will use the convention that $f_\theta(x_1|x_0) = f_\theta(x_1)$.

Given an observed sequence $\{Y_1, \ldots, Y_{MT}\}$ of length MT, we will split this sequence into M subsequences, each of length T. Our objective is to obtain unbiased estimate for the likelihood function

$$\eta(\theta) := \int p_\theta(x_{1:MT}, Y_{1:MT})\, dx_{1:MT}$$

and $\psi_{MT} := \mathbb{E}_p[\psi(X_{1:MT})|Y_{1:MT}]$ where $\mathbb{E}_p$ denotes the expectation taken under the HMM (3.1.1).

Define for $1 \leq m \leq M$ and $1 \leq t \leq T$, $Y_{m,t} := Y_{(m-1)T+t}$ and $Y_{m,1:T} := \{Y_{m,t}, 1 \leq t \leq T\}$. Similarly, we will define $X_{m,t} := X_{(m-1)T+t}$ and $X_{m,1:T} := \{X_{m,t}, 1 \leq t \leq T\}$. Note that for each m, $Y_{m,1}, \ldots, Y_{m,T}$ are conditionally independent given $X_{m,1:T}$. Since the data is segmented, for the m-th sequence, the distribution of $X_{m,1}$ may not have

a simple form. As such, one will need to choose a distribution $f_{\theta m}$ to generate the sequence for the purpose of computing the weights. With that, for $1 \leq t \leq T$,

$$X_{m,1} \sim f_{\theta m}(\cdot), \quad X_{m,t} \sim f_{\theta}(\cdot|X_{m,t-1}), \quad Y_{m,t} \sim g_{\theta}(\cdot|X_{m,t}).$$

For the m-th data sequence, we define

$$p_{\theta m}(x_{m,1:T}, Y_{m,1:T}) := \prod_{t=1}^{T} f_{\theta}(x_{m,t}|x_{m,t-1}) \prod_{t=1}^{T} g_{\theta}(Y_{m,t}|x_{m,t}),$$

where $f_{\theta}(x_{m,1}|x_{m,0}) := f_{\theta m}(x_{m,1})$.

For the parallel particle filter set-up, we will construct M particle filters using K particles for each filter, running in parallel, for each subsequence. For the m-th subsequence, we shall let $q_{m,t}(\cdot|x_{m,1:t-1})$ be the proposal density with the convention that $q_{m,1}(\cdot|x_{m,1:0}) = q_{m,1}(\cdot)$. The pseudo code for the m-th particle filter is given in Algorithm 11. In this algorithm, we will perform resampling at each time t for $1 \leq t \leq T$ using a multinomial resampling scheme. We shall also track the ancestral origins of the particles in this algorithm.

After we have run the algorithm, we will have M strings of K particles $\{X^k_{m,1:T}\}$ with associated weights $\left\{\prod_{t=1}^{T} w_{m,t}(\tilde{X}^k_{m,1:t})\right\}$ that we can use for the estimation of the likelihood function. Since the M particle filters are run in parallel, to make use of all these particles, we will create a "coalesced" particle using the k_m particle from the m-th subsequence, running k_m from 1 to K. Specifically, the coalesced particle will be of the form

$$(X^{k_1}_{1,1:T}, X^{k_2}_{2,1:T}, \ldots, X^{k_M}_{M,1:T})$$

where $1 \leq k_m \leq K$ for $1 \leq m \leq M$. Accordingly, we will have a total of K^M coalesced particles for the computation of the estimate. Since the convergence of the particle filter are well established, one will just need to account for the link between the subsequences to ensure that the proposed method will be able to provide estimation for smoothing and filtering. We shall give the details in the following section.

Algorithm 11: m-th Particle Filter

for $t = 1 : T$ **do**

for $k = 1 : K$ **do**

- Importance Sampling: Draw an i.i.d sample $\tilde{X}^k_{m,t} \sim q_{m,t}(\cdot|X^k_{m,1:t-1})$ and set $\tilde{X}^k_{m,1:t} = (X_{m,1:t-1}, \tilde{X}^k_{m,t})$.

- Compute weights:

$$\begin{aligned} w_{m,1}(\tilde{X}^k_{m,1}) &= \frac{f_{\theta m}(\tilde{X}^k_{m,1}) g_\theta(Y_{m,1}|\tilde{X}^k_{m,1})}{q_{m,1}(\tilde{X}^k_{m,1})}, \\ w_{m,t}(\tilde{X}^k_{m,1:t}) &= \frac{f_\theta(\tilde{X}^k_{m,t}|X^k_{m,t-1}) g_\theta(Y_{m,t}|\tilde{X}^k_{m,t})}{q_{m,t}(\tilde{X}^k_{m,t}|X^k_{m,1:t-1})} \end{aligned}$$

  and let $\bar{w}_{m,t} = K^{-1} \sum_{k=1}^K w_{m,t}(\tilde{X}^k_{m,1:t})$.

- Resampling: Generate i.i.d. $B^1_{m,t}, \ldots, B^K_{m,t}$ such that

$$P(B^k_{m,t} = j) = w_{m,t}(\tilde{X}^j_{m,1:t})/K\bar{w}_{m,t}.$$

- Updating: Set $(X^k_{m,1:t}, A^k_{m,t}) = (\tilde{X}^{B^k_{m,t}}_{m,1:t}, A^{B^k_{m,t}}_{m,t-1})$ where $A^k_{m,0} = k$.

end

end

3.2 Proposed estimates

In this section, we will first introduce some notations and definitions for the purpose of defining the proposed estimate for ψ_{MT} and the likelihood function $\eta(\theta)$. We begin by stating some notations that will be used throughout this chapter.

Definition 3.2.1. The reciprocal of the product of the weights of the k-particle up to time t of the m-th subsequence is denoted by

$$h_{m,t}(x^k_{m,1:t}) = \left[\prod_{u=1}^{t} w_{m,u}(x_{m,1:u})\right]^{-1},$$

with the convention that $h_{m,0} \equiv 1$.

Definition 3.2.2. The mean weight of the particles at time t before resampling for the m-th subsequence is defined as

$$\bar{w}_{m,t} = \frac{1}{K}\sum_{k=1}^{K} w_{m,t}(\tilde{x}^k_{m,1:t}).$$

Definition 3.2.3. The adjusted weights for the k-th particle before and after resampling are defined by

$$\tilde{H}^k_{m,t} = \prod_{u=1}^{t} \frac{\bar{w}_{m,u}}{w_{m,u}(\tilde{x}^k_{m,1:u})} \text{ and } H^k_{m,t} = \prod_{u=1}^{t} \frac{\bar{w}_{m,u}}{w_{m,u}(x^k_{m,1:u})},$$

respectively with the convention that $H^k_{m,0} = 1$.

For our proposed method, we have split the observed sequence and run the particle filters in parallel. In doing so, we have ignored the dependency of the subsequences. To rectify this, we have to account for the dependency between the sequences of latent states obtained via the parallel particle filters. In particular, if we have chosen the k_m particle from the m-th subsequence, we will have to introduce a weight function between the hidden states $x^{k_{m-1}}_{m-1,T}$ and $x^{k_m}_{m,1}$ for $2 \leq m \leq M$.

Definition 3.2.4. We define the link weight between the k_{m-1} particle from the $(m-1)$-th sequence and the k_m particle from the m-th sequence by

$$\alpha_m(x^{k_{m-1}}_{m-1,T}, x^{k_m}_{m,1}) := \frac{f_\theta(x^{k_m}_{m,1}|x^{k_{m-1}}_{m-1,T})}{f_{\theta m}(x^{k_m}_{m,1})}.$$

3.2.1 Estimate for likelihood function

Using the above notations, we shall now give an expression for the proposed estimate for the likelihood $\eta(\theta)$ under a parallel particle filter set-up.

Definition 3.2.5. Let $\mathbf{k} = (k_1, \ldots, k_M)$, $\mathbf{1} = (1, \ldots, 1)$ and $\mathbf{K} = (K, \ldots, K)$. We define

$$\hat{\eta}(\theta) := \left[\prod_{m=1}^{M}\prod_{t=1}^{T} \bar{w}_{m,t}\right]\left[\frac{1}{K^M}\sum_{\mathbf{k}:\mathbf{1}\prec\mathbf{k}\prec\mathbf{K}}\left(\prod_{m=2}^{M}\alpha_m(X_{m-1,T}^{k_{m-1}}, X_{m,1}^{k_m})\right)\right] \tag{3.2.1}$$

where $\mathbf{i} \prec \mathbf{j}$ means that $i_m \leq j_m$ for all $1 \leq m \leq M$.

As a result, when we run the algorithm, we will have to compute the sum of link weights for the calculation of the unbiased estimate of the likelihood. The pseudo code for the parallel particle filter is given in Algorithm 12.

Let $X_{1:MT}^{\mathbf{k}} = (X_{1,1:T}^{k_1}, \ldots, X_{M,1:T}^{k_M})$ and $p_\theta(x_{1:MT}, y_{1:MT}) = \prod_{t=1}^{MT} f_\theta(x_t|x_{t-1})g_\theta(y_t|x_t)$ with the convention $f_\theta(\cdot|x_0) = f_{\theta 1}(\cdot)$ and $q_{m,T}(x_{m,1:T}) = \prod_{t=1}^{T} q_{m,t}(x_{m,t}|x_{m,1:t-1})$ with the convention that $q_{m,1}(\cdot|x_{m,0}) = q_{m,1}(\cdot)$.

Definition 3.2.6. The likelihood between the actual density and the simulated density is defined as

$$L(x_{1:MT}^{\mathbf{k}}) = \frac{p_\theta(x_{1:MT}^{\mathbf{k}}, y_{1:MT})}{\prod_{m=1}^{M} q_{m,T}(x_{m,1:T}^{k_m})}.$$

For the purpose of our proof for the unbiasedness of $\hat{\eta}(\theta)$, we shall use an alternative expression of $\hat{\eta}(\theta)$ in terms of L.

Lemma 3.1

The estimate $\hat{\eta}(\theta)$ can be expressed as

$$\hat{\eta}(\theta) = \frac{1}{K^M}\sum_{\mathbf{k}:\mathbf{1}\prec\mathbf{k}\prec\mathbf{K}} L(X_{1:MT}^{\mathbf{k}})H_{MT}^{\mathbf{k}} \tag{3.2.2}$$

where $H_{MT}^{\mathbf{k}} := \prod_{m=1}^{M} H_{m,T}^{k_m}$.

Algorithm 12: Parallel Particle Filter

begin

Initialisation;

for $m = 1 : M$ **do**

for $k = 1 : K$ **do**

Generate $\tilde{X}^k_{m,1} \sim q_{m,1}(\cdot)$ and compute weights $w_{m,1}(\tilde{X}^k_{m,1})$.

Store $\bar{w}_{m,1}$ into matrix **Wbar**.

Compute normalised weights $W_{m,1}(\tilde{X}^k_{m,1})$ and perform resampling using a multinomial distribution.

Store $w_{m,1}(X^k_{m,1})$ into matrix **W**.

end

end

for $m = 1 : M$ **do**

for $t = 2 : T$ **do**

for $k = 1 : N$ **do**

Generate $\tilde{X}^k_{m,t} \sim q_{m,t}(\cdot|X^k_{m,1:t-1})$ and compute weights $w_{m,t}(\tilde{X}^k_{m,t})$.

Store $\bar{w}_{m,t}$ into matrix **Wbar**.

Compute normalised weights $W_{m,t}(\tilde{X}^k_{m,t})$ and perform resampling using a multinomial distribution.

Store $w_{m,1}(X^k_{m,t})$ into matrix **W**.

end

end

end

for $m = 1 : M - 1$ **do**

for $k_m = 1 : K$ **do**

Compute $\alpha_{m+1}(X^{k_m}_{m,T}, X^{k_{m+1}}_{m+1,1})$

end

end

end

Proof:

Note that

$$\begin{aligned} L(X^{\mathbf{k}}_{1:MT}) &= \frac{\prod_{m=1}^{M} p_\theta(x^{k_m}_{m,1:T}, y_{m,1:T})}{\prod_{m=1}^{M} q_{m,T}(x^{k_m}_{m,1:T})} \\ &= \prod_{m=1}^{M} \frac{p_\theta(x^{k_m}_{m,1:T}, y_{m,1:T})}{q_{m,T}(x^{k_m}_{m,1:T})} \\ &= \prod_{m=1}^{M} \frac{p_{\theta m}(x^{k_m}_{m,1:T}, y_{m,1:T})}{q_{m,T}(x^{k_m}_{m,1:T})} \prod_{m=1}^{M} \frac{p_\theta(x^{k_m}_{m,1})}{p_{\theta m}(x^{k_m}_{m,1})} \\ &= \left(\prod_{m=1}^{M} \prod_{t=1}^{T} w_{m,t}(x^{k_m}_{m,1:t}) \right) \prod_{m=2}^{M} \alpha_m(x^{k_{m-1}}_{m-1,T}, x^{k_m}_{m,1}). \end{aligned}$$

Thus,

$$\frac{L(X^{\mathbf{k}}_{1:MT})}{\prod_{m=1}^{M} \prod_{t=1}^{T} \bar{w}_{m,t}} = \prod_{m=1}^{M} \left(H^{k_m}_{m,T}\right)^{-1} \prod_{m=2}^{M} \alpha_{m,T}(x^{k_{m-1}}_{m-1,T}, x^{k_m}_{m,1}). \tag{3.2.3}$$

By rearranging the terms in (3.2.3) and substituting into (3.2.1), the result follows.

□

3.2.2 Estimate involving latent states

For the estimation of ψ_{MT}, we will need to use the conditional density of $X_{1:MT}$ given $Y_{1:MT}$ which is given by

$$p_\theta(x_{1:MT}|Y_{1:MT}) = \prod_{t=1}^{MT} f_\theta(x_t|x_{t-1}) g_\theta(Y_t|x_t)/\eta(\theta). \tag{3.2.4}$$

Definition 3.2.7. The likelihood between the conditional density and the simulated density is defined as

$$L_c(X^{\mathbf{k}}_{1:MT}) := \frac{p_\theta(x^{\mathbf{k}}_{1:MT}|Y_{1:MT})}{\prod_{m=1}^{M} q_{m,T}(x^{k_m}_{m,1:T})} = L(X^{\mathbf{k}}_{1:MT})/\eta(\theta). \tag{3.2.5}$$

Canonical case

For the canonical case, $\eta(\theta)$ is known. We proposed the following estimate for ψ_{MT} which we will show is unbiased.

Definition 3.2.8. The estimate of ψ_{MT} for the canonical case is defined by

$$\hat{\psi}_{MT} := \frac{1}{K^M} \sum_{\mathbf{k} \in \mathbb{Z}^M} L_c(X_{1:MT}^{\mathbf{k}}) H_{MT}^{\mathbf{k}} \psi(X_{1:MT}^{\mathbf{k}}) \tag{3.2.6}$$

where $H_{MT}^{\mathbf{k}} = \prod_{m=1}^{M} H_{m,T}^{k_m}$.

By (3.2.4) and (3.2.5), $\eta(\theta)$ appears on the RHS of (3.2.6). Consequently, if we set $\psi \equiv 1$, we will have the estimate $\hat{\eta}(\theta)$. With this in mind, once we have proven that $\hat{\psi}_{MT}$ is unbiased, the unbiasedness of $\hat{\eta}(\theta)$ will follow.

Non-canonical case

For the non-canonical case, we will estimate the unknown $\eta(\theta)$ using the unbiased estimate $\hat{\eta}(\theta)$. As a consequence, we have the following estimate for ψ_{MT}.

Definition 3.2.9. For the non-canonical case, an estimate for ψ_{MT} is given by

$$\tilde{\psi}_{MT} = \frac{\sum_{\mathbf{k} \in \mathbb{Z}^M} L(X_{1:MT}^{\mathbf{k}}) \psi(X_{1:MT}^{\mathbf{k}}) H_{MT}^{\mathbf{k}}}{\sum_{\mathbf{k} \in \mathbb{Z}^M} L(X_{1:MT}^{\mathbf{k}}) H_{MT}^{\mathbf{k}}}. \tag{3.2.7}$$

We remark that this estimate is asymptotically unbiased. If we set $\psi(X_{1:MT}) = X_t$ where $1 \leq t \leq MT$, we will be able to obtain estimates for the smoothed means, which are useful in applications.

3.2.3 Technical lemma

To prove that the proposed estimate is unbiased for ψ_{MT} in the canonical case, we will make use of martingale differences. To do this, we will express $\hat{\psi}_{MT}$ using martingale differences. The following lemma is useful in establishing the martingale difference expression of $\hat{\psi}_{MT}$.

Lemma 3.2

The adjusted weights satisfies the following relations:

(i)

$$\tilde{H}_{m,t}^{k} = H_{m,t-1}^{k} \frac{\bar{w}_{m,t}}{w_{m,t}(\tilde{X}_{m,t}^{k})}. \tag{3.2.8}$$

(ii)

$$\sum_{k=1}^{K} H_{m,t}^{k} = \sum_{k=1}^{K} \tilde{H}_{m,t}^{k} \#_{t}^{k}, \tag{3.2.9}$$

where $(\#_t^1, \ldots, \#_t^K) \sim$ *Multinomial*$(K, W_n^{1:K})$ *with* $W_n = w_{m,u}(\tilde{X}_{m,1:u}^{n})/(m\bar{w}_{m,u})$ *are the number of copies of* $\tilde{X}_{m,1:u}^{1}, \ldots, \tilde{X}_{m,1:u}^{K}$ *obtained from resampling at time* t.

Proof:

(i) Note that $H_{m,t}^{k} = \left(\prod_{u=1}^{t} \bar{w}_{m,u}\right) h_{m,t}(x_{m,1:t}^{k})$ is computed using the particles from resampling at time t and $\tilde{H}_{m,t}^{k}$ is computed using the k-th particle before resampling at time t for the m-th sequence. At time t before resampling, $\tilde{H}_{m,t}^{k}$ is computed by multiplying $\bar{w}_{m,t}/w_{m,t}(\tilde{x}_{m,t}^{k})$ to $H_{m,t-1}^{k}$ (weight after resampling is done at time $t-1$) and result follows.

(ii) At time t, after resampling, we will have $\#_t^k$ copies of $\tilde{X}_{m,1:u}^{k}$ for $1 \leq k \leq K$. Since all these copies form the final sample after resampling is done, by taking the sum for all K particles, the result follows.

□

3.3 Main theorem

Before we proceed with the proof of the unbiasedness property of the proposed estimate of ψ_{MT} under the canonical case, we will first establish the martingale difference expression for $\hat{\psi}_{MT}$.

Suppose the sequence of length MT is generated using $q_{m,t}$ (without data segmentation) for $1 \leq m \leq M$ and $1 \leq t \leq T$ independently on the M segments without resampling. Let $\mathbb{E}_q(\cdot)$ denotes the expectation taken with respect to this particle filter set-up. Let $\xi_t(x_{1:t}) = \mathbb{E}_q[L_c(X_{1:MT})\psi(X_{1:MT})|X_{1:t} = x_{1:t}]$. We construct a filtration $\{\mathcal{F}_u : 1 \leq u \leq 2MT\}$, with $\mathcal{F}_{2t-1}$ denoting the σ-algebra for all random variables generated up to time t before resampling is done and $\mathcal{F}_{2t}$ denoting the σ-algebra for all random variables generated up to time t after resampling is done. For $t = (m-1)T + u$, define

$$\epsilon_{2t-1}^k := K^{M-m} \sum_{\mathbf{k}_{m-1}:\mathbf{1}\prec\mathbf{k}_{m-1}\prec\mathbf{K}} \left(\prod_{r=1}^{m-1} H_{r,T}^{k_r}\right) H_{m,u-1}^k [\xi_t(\tilde{X}_{1:t}^{(\mathbf{k}_{m-1},k)}) - \xi_{t-1}(X_{1:t-1}^{(\mathbf{k}_{m-1},k)})] \tag{3.3.1}$$

where $\mathbf{k}_{m-1} = (k_1, \ldots, k_{m-1})$ and the convention that when $u = 1$, $X_{1:t-1}^{(\mathbf{k}_{m-1},k)} = X_{1:t-1}^{\mathbf{k}_{m-1}}$. We also define

$$\epsilon_{2t}^k := K^{M-m} \left[\sum_{\mathbf{k}_{m-1}:\mathbf{1}\prec\mathbf{k}_{m-1}\prec\mathbf{K}} \left(\prod_{r=1}^{m-1} H_{r,T}^{k_r}\right) \tilde{H}_{m,u}^k \xi_t(\tilde{X}_{1:t}^{(\mathbf{k}_{m-1},k)})\right] \left[\#_t^k - w_{m,u}(\tilde{X}_{m,u}^k)/\bar{w}_{m,u}\right], \tag{3.3.2}$$

where $(\#_t^1, \ldots, \#_t^K) \sim \text{Multinomial}(K, W_n^{1:K})$ with $W_n = w_{m,u}(\tilde{X}_{m,1:u}^n)/(m\bar{w}_{m,u})$ are the number of copies of $\tilde{X}_{m,1:u}^1, \ldots, \tilde{X}_{m,1:u}^K$ in the sample obtained from resampling at time t.

Lemma 3.3

The estimate $\hat{\psi}_{MT}$ can be expressed as

$$\hat{\psi}_{MT} = \psi_{MT} + \frac{1}{K^M} \sum_{s=1}^{2MT} \sum_{k=1}^{K} \epsilon_s^k. \tag{3.3.3}$$

Proof: (i) Case 1: $M = 1$

When $M = 1$, there is only 1 sequence to deal with. Thus, we will just need to ensure that the required sum is indeed telescoping within this sequence. We need to show

$$\hat{\psi}_{MT} - \psi_{MT} = \frac{1}{K} \sum_{s=1}^{2T} \sum_{k=1}^{K} \epsilon_s^k$$

where

$$\begin{aligned}\epsilon_{2t-1}^k &= H_{t-1}^k \left[\xi_t(\tilde{X}_{1:T}^k) - \xi_{t-1}(X_{1:t-1}^k)\right], \\ \epsilon_{2t}^k &= \tilde{H}_t^k \xi_t(\tilde{X}_{1:t}^k) \left[\#_t^k - w_t(\tilde{X}_t^k)/\bar{w}_t\right].\end{aligned}$$

For this case, we will ignore the use of $m = 1$ in the subscripts to simplify the expression. Consider the first term of ϵ_{2t-1}^k in (3.3.1) and the second term of ϵ_{2t}^k in (3.3.2). By (3.2.8),

$$\begin{aligned}H_{t-1}^k \xi_t(\tilde{X}_{1:t}^k) - \tilde{H}_t^k \xi_t(\tilde{X}_{1:t}^k) w_t(\tilde{X}_t^k)/\bar{w}_t &= \xi_t(\tilde{X}_{1:t}^k) \left[H_{t-1}^k - \tilde{H}_t^k \cdot w_t(\tilde{X}_t^k)/\bar{w}_t\right] \\ &= 0.\end{aligned}$$

Next, consider the first term of $\sum_{k=1}^K \epsilon_{2t}^k$ by summing all particles given by (3.3.2) and the second term of $\sum_{k=1}^K \epsilon_{2t+1}^k$ by summing all particles given by (3.3.1). Using (3.2.9), we have

$$\begin{aligned}\sum_{k=1}^K \tilde{H}_t^k \xi_t(\tilde{X}_{1:t}^k)\#_t^k - \sum_{k=1}^K H_t^k \xi_t(X_{1:t}^k) &= \sum_{k=1}^K H_t^k \xi_t(X_{1:t}^k) - \sum_{k=1}^K H_t^k \xi_t(X_{1:t}^k) \\ &= 0.\end{aligned}$$

Note that $\xi_0(X_0) = \psi_{MT}$ and $\xi_T(X_{1:T}^k) = L_c(X_{1:T}^k)\psi(X_{1:T}^k)$. Using this and the previous two equations, we have shown that $\frac{1}{K}\sum_{s=1}^{2T}\sum_{k=1}^K \epsilon_s^k$ is indeed a telescoping series that reduces to $\hat{\psi}_{MT} - \psi_{MT}$ and the case for $M = 1$ is proven.

(ii) Case 2: $M = 2$

When $M = 2$, we have to deal with 2 subsequences. We need to show

$$\hat{\psi}_{MT} - \psi_{MT} = \frac{1}{K^2}\sum_{s=1}^{4T}\sum_{k=1}^K \epsilon_s^k$$

where for $1 \le t \le T$,

$$\begin{aligned}\epsilon_{2t-1}^k &= K H_{t-1}^k \left[\xi_t(\tilde{X}_{1:T}^k) - \xi_{t-1}(X_{1:t-1}^k)\right], \\ \epsilon_{2t}^k &= K \tilde{H}_t^k \xi_t(\tilde{X}_{1:t}^k) \left[\#_t^k - w_t(\tilde{X}_t^k)/\bar{w}_t\right].\end{aligned}$$

and for $t = T + u$ where $1 \le u \le T$,

$$\begin{aligned}\epsilon_{2t-1}^k &= \sum_{k_1=1}^K H_{1,T}^{k_1} H_{2,u-1}^k \left[\xi_t(\tilde{X}_{1:t}^{(k_1,k)}) - \xi_{t-1}(X_{1:t-1}^{(k_1,k)})\right], \\ \epsilon_{2t}^k &= \sum_{k_1=1}^K H_{1,T}^{k_1} \tilde{H}_{2,u}^k \xi_t(\tilde{X}_{1:t}^{(k_1,k)}) \left[\#_t^k - w_{m,u}(\tilde{X}_{m,u}^k)/\bar{w}_{m,u}\right].\end{aligned}$$

Using the result for $M = 1$, the associated sum for each subsequence is telescoping and reduces to

$$\begin{aligned}
\frac{1}{K^2}\sum_{s=1}^{2T}\sum_{k=1}^{K}\epsilon_s^k &= \frac{1}{K}\sum_{k=1}^{K}\tilde{H}_T^k\xi_T(\tilde{X}_{1:T}^k)\#_T^k - \psi_{MT}, \\
\frac{1}{K^2}\sum_{s=2T+1}^{4T}\sum_{k=1}^{K}\epsilon_s^k &= \hat{\psi}_{MT} - \frac{1}{K^2}\sum_{k=1}^{K}\sum_{k_1=1}^{K}H_{1,T}^{k_1}\xi_T(X_{1:T}^{k_1}) \\
&= \hat{\psi}_{MT} - \frac{1}{K}\sum_{k_1=1}^{K}H_{1,T}^{k_1}\xi_T(X_{1:T}^{k_1}).
\end{aligned}$$

Changing the dummy variable in the relevant sums and using (3.2.9), we have

$$\begin{aligned}
\frac{1}{K^2}\sum_{s=1}^{4T}\sum_{k=1}^{K}\epsilon_s^k &= \frac{1}{K^2}\sum_{s=1}^{2T}\sum_{k=1}^{K}\epsilon_s^k + \frac{1}{K^2}\sum_{s=2T+1}^{4T}\sum_{k=1}^{K}\epsilon_s^k \\
&= \hat{\psi}_{MT} - \frac{1}{K}\sum_{k_1=1}^{K}H_{1,T}^{k_1}\xi_T(X_{1:T}^{k_1}) + \frac{1}{K}\sum_{k=1}^{K}\tilde{H}_T^k\xi_T(\tilde{X}_{1:T}^k)\#_T^k - \psi_{MT} \\
&= \hat{\psi}_{MT} - \frac{1}{K}\sum_{i=1}^{K}H_{1,T}^{i}\xi_T(X_{1:T}^{i}) + \frac{1}{K}\sum_{i=1}^{K}\tilde{H}_T^i\xi_T(\tilde{X}_{1:T}^i)\#_T^k - \psi_{MT} \\
&= \hat{\psi}_{MT} - \frac{1}{K}\sum_{i=1}^{K}H_{1,T}^{i}\xi_T(X_{1:T}^{i}) + \frac{1}{K}\sum_{i=1}^{K}H_T^i\xi_T(X_{1:T}^i) - \psi_{MT} \\
&= \hat{\psi}_{MT} - \psi_{MT},
\end{aligned}$$

proving the result for this case.

(iii) Case 3: $M > 2$

Note that in the proof for the case of $M = 2$, the key part is to show that the associated terms for the link between the first and second subsequence cancels one another. Repeating this argument, it suffices to show that the terms associated with the link between m-th and $(m+1)$-th subsequences cancels for $1 \leq m \leq M-1$. To be exact, we need to show that the first term of $\sum_{k=1}^{K}\epsilon_{(m-1)T}^k$ and the second term of $\sum_{k=1}^{K}\epsilon_{(m-1)T+1}^k$ are equal. The first term of $\sum_{k=1}^{K}\epsilon_{(m-1)T}^k$ is

$$K^{M-m}\sum_{k=1}^{K}\sum_{k_1=1}^{K}\cdots\sum_{k_{m-1}=1}^{K}\left(\prod_{r=1}^{m-1}H_{r,T}^{k_r}\right)\tilde{H}_{m,T}^k\xi_{(m-1)T}(\tilde{X}_{1:(m-1)T}^{(\mathbf{k}_{m-1},k)})\#_{(m-1)T}^k. \quad (3.3.4)$$

Rewriting the second term of $\sum_{k=1}^{K} \epsilon^k_{(m-1)T+1}$, we have
$K^{M-m-1} \sum_{k=1}^{K} \sum_{\mathbf{k}_m:\mathbf{1}\prec\mathbf{k}_m\prec\mathbf{K}} \left(\prod_{r=1}^{m} H^{k_r}_{r,T}\right) \xi_{(m-1)T}(X^{\mathbf{k}_m}_{1:(m-1)T})$

$$\begin{aligned} &= K^{M-m-1} \cdot K \sum_{\mathbf{k}_m:\mathbf{1}\prec\mathbf{k}_m\prec\mathbf{K}} \left(\prod_{r=1}^{m} H^{k_r}_{r,T}\right) \xi_{(m-1)T}(X^{\mathbf{k}_m}_{1:(m-1)T}) \\ &= K^{M-m} \sum_{\mathbf{k}_m:\mathbf{1}\prec\mathbf{k}_m\prec\mathbf{K}} \left(\prod_{r=1}^{m} H^{k_r}_{r,T}\right) \xi_{(m-1)T}(X^{\mathbf{k}_m}_{1:(m-1)T}) \end{aligned}$$

Changing the dummy variable for the sum in (3.3.4) and using (3.2.9), we have

$$\begin{aligned} &\sum_{k=1}^{K}\sum_{k_1=1}^{K} \cdots \sum_{k_{m-1}=1}^{K} \left(\prod_{r=1}^{m-1} H^{k_r}_{r,T}\right) \tilde{H}^{k}_{m,T} \xi_{(m-1)T}(\tilde{X}^{(\mathbf{k}_{m-1},k)}_{1:(m-1)T}) \#^{k}_{(m-1)T} \\ &= \sum_{k_m=1}^{K}\sum_{k_1=1}^{K} \cdots \sum_{k_{m-1}=1}^{K} \left(\prod_{r=1}^{m-1} H^{k_r}_{r,T}\right) \tilde{H}^{k_m}_{m,T} \xi_{(m-1)T}(\tilde{X}^{(\mathbf{k}_{m-1},k_m)}_{1:(m-1)T}) \#^{k_m}_{(m-1)T} \\ &= \sum_{k_m=1}^{K}\sum_{k_1=1}^{K} \cdots \sum_{k_{m-1}=1}^{K} \left(\prod_{r=1}^{m-1} H^{k_r}_{r,T}\right) H^{k_m}_{m,T} \xi_{(m-1)T}(X^{\mathbf{k}_m}_{1:(m-1)T}) \\ &= \sum_{k_1=1}^{K} \cdots \sum_{k_{m-1}=1}^{K}\sum_{k_m=1}^{K} \left(\prod_{r=1}^{m} H^{k_r}_{r,T}\right) \xi_{(m-1)T}(X^{\mathbf{k}_m}_{1:(m-1)T}) \\ &= \sum_{\mathbf{k}_m:\mathbf{1}\prec\mathbf{k}_m\prec\mathbf{K}} \left(\prod_{r=1}^{m} H^{k_r}_{r,T}\right) \xi_{(m-1)T}(X^{\mathbf{k}_m}_{1:(m-1)T}). \end{aligned}$$

As a result, the first term of $\sum_{k=1}^{K} \epsilon^k_{(m-1)T}$ and the second term of $\sum_{k=1}^{K} \epsilon^k_{(m-1)T+1}$ are equal for $2 \leq m \leq M$ and result follows.

□

We will now state and prove the main theorem of this chapter.

Theorem 3.4

Let $\theta \in \Theta$ and $\psi_{MT} < \infty$. Then $\hat{\psi}_{MT}$ is an unbiased estimator for ψ_{MT}.

Proof:

We will prove this theorem using a martingale difference argument. Using Lemma 3.3, we have

$$\hat{\psi}_{MT} - \psi_{MT} = \frac{1}{K} \sum_{s=1}^{2MT} \sum_{k=1}^{K} \epsilon^k_s.$$

Using tower property from Theorem 1.1, we have

$$\begin{aligned}\mathbb{E}[\xi_t(x_{1:t})|\mathcal{F}_{2t-2}] &= \mathbb{E}\left[\mathbb{E}_q[L_c(X_{1:MT})\psi(X_{1:MT})|X_{1:t}=x_{1:t}]|\mathcal{F}_{2t-2}\right] \\ &= \mathbb{E}_q[L_c(X_{1:MT})\psi(X_{1:MT})|X_{1:t-1}=x_{1:t-1}] \\ &= \xi_{t-1}(x_{1:t-1})\end{aligned}$$

where $\mathbb{E}$ is the expectation taken with respect to the probability associated to the parallel particle filter set-up. Thus taking conditional expectation on (3.3.1), we have

$$\begin{aligned}\mathbb{E}[\epsilon_{2t-1}^k|\mathcal{F}_{2t-2}] &= \sum_{\mathbf{k}_{m-1}:\mathbf{1}\prec\mathbf{k}_{m-1}\prec\mathbf{K}} \left(\prod_{r=1}^{m-1} H_{r,T}^{k_r}\right) \frac{H_{m,u-1}^k}{K^{m-M}} \mathbb{E}[\xi_t(\tilde{X}_{1:t}^{(\mathbf{k}_{m-1},k)}) - \xi_{t-1}(X_{1:t-1}^{(\mathbf{k}_{m-1},k)})|\mathcal{F}_{2t-2}] \\ &= \sum_{\mathbf{k}_{m-1}:\mathbf{1}\prec\mathbf{k}_{m-1}\prec\mathbf{K}} \left(\prod_{r=1}^{m-1} H_{r,T}^{k_r}\right) \frac{H_{m,u-1}^k}{K^{m-M}} [\xi_{t-1}(\tilde{X}_{1:t-1}^{(\mathbf{k}_{m-1},k)}) - \xi_{t-1}(X_{1:t-1}^{(\mathbf{k}_{m-1},k)})] \\ &= 0.\end{aligned}$$

Under the parallel particle filter set-up, for $t=(m-1)T+u$, we have

$$\begin{aligned}\mathbb{E}[\#_t^k|\mathcal{F}_{2t-1}] &= K\cdot\frac{w_{m,u}(\tilde{X}_{m,1:u}^k)}{K\bar{w}_{m,u}} \\ &= \frac{w_{m,u}(\tilde{X}_{m,1:u}^k)}{\bar{w}_{m,u}}.\end{aligned}$$

Hence, taking conditional expectation on (3.3.2), we have

$$\begin{aligned}\mathbb{E}[\epsilon_{2t}^k|\mathcal{F}_{2t-1}] &= K^{M-m}\left[\sum_{\mathbf{k}_{m-1}:\mathbf{1}\prec\mathbf{k}_{m-1}\prec\mathbf{K}} \left(\prod_{r=1}^{m-1} H_{r,T}^{k_r}\right) \tilde{H}_{m,u}^k \xi_t(\tilde{X}_{1:t}^{(\mathbf{k}_{m-1},k)})\right] \\ &\quad \times\mathbb{E}\left[\#_t^k - w_{m,u}(\tilde{X}_{m,u}^k)/\bar{w}_{m,u}|\mathcal{F}_{2t-1}\right] \\ &= 0.\end{aligned}$$

Therefore $\{\epsilon_s^k\}_{s=1}^{2MT}$ is a sequence of martingale difference for each $k\in\{1,\ldots,K\}$. Applying the tower property (Theorem 1.1) on $E[\hat{\psi}_{MT}-\psi_{MT}]$, we have

$$\begin{aligned}\mathbb{E}[\hat{\psi}_{MT}-\psi_{MT}] &= \frac{1}{K^M}\sum_{s=1}^{2MT}\sum_{k=1}^{K}\mathbb{E}[\epsilon_s^k] \\ &= \frac{1}{K^M}\sum_{t=1}^{MT}\sum_{k=1}^{K}\mathbb{E}[\epsilon_{2t-1}^k] + \frac{1}{K^M}\sum_{t=1}^{MT}\sum_{k=1}^{K}\mathbb{E}[\epsilon_{2t}^k] \\ &= \frac{1}{K^M}\sum_{t=1}^{MT}\sum_{k=1}^{K}\mathbb{E}[\mathbb{E}[\epsilon_{2t-1}^k|\mathcal{F}_{2t-2}]] + \frac{1}{K^M}\sum_{t=1}^{MT}\sum_{k=1}^{K}\mathbb{E}[\mathbb{E}[\epsilon_{2t}^k|\mathcal{F}_{2t-1}]] \\ &= 0.\end{aligned}$$

As a consequence, $\mathbb{E}[\hat{\psi}_{MT}] = \psi_{MT}$. □

Corollary 3.5

Let $\theta \in \Theta$ and $\eta(\theta) < \infty$. Then $\hat{\eta}(\theta)$ is an unbiased estimator for $\eta(\theta)$.

Proof:

Since $L_c(X_{1:MT}) = L(X_{1:MT})/\eta(\theta)$, setting $\psi \equiv 1$ in equation (3.2.6) and using Theorem 1.1, we have $\mathbb{E}[\hat{\eta}(\theta)/\eta(\theta)] = 1$. Therefore $\mathbb{E}[\hat{\eta}(\theta)] = \eta(\theta)$, completing the proof. □

3.4 Ancestral origin representation

To obtain approximation for the standard errors of the proposed estimates, one needs another martingale difference expression involving the ancestral origins. In this section, we shall introduce this expression and prove the unbiasedness property.

We shall make use of the following notations for this section:

(i) $\tilde{X}^{\mathbf{k}}_{1:t} = (X^{k_1}_{1,T}, \ldots, X^{k_{m-1}}_{m-1,T}, \tilde{X}^{k_m}_{m,u})$ and $X^{\mathbf{k}}_{1:t} = (X^{k_1}_{1,T}, \ldots, X^{k_{m-1}}_{m-1,T}, X^{k_m}_{m,u})$ where $t = (m-1)T + u$ for $1 \le u \le T$,

(ii) $\tilde{H}^{\mathbf{k}}_t = \left(\prod_{r=1}^{m-1} H^{k_r}_{r,T}\right) \tilde{H}^{k_m}_{m,T}$,

(iii) $H^{\mathbf{k}}_t = \left(\prod_{r=1}^{m-1} H^{k_r}_{r,T}\right) H^{k_m}_{m,T}$.

To emphasise the position of the subsequence, we shall let $k_m = k(m)$.

Theorem 3.6

Define, for $(m-1)T < t \le mT$ and $1 \le j \le K$,

$$\begin{aligned} \epsilon^j_{2t-1} &= K^{-m+1} \sum_{\mathbf{k}\in\mathbb{Z}^m : A^{k(m)}_{m,t-1}=j} [\xi_t(\tilde{X}^{\mathbf{k}}_{1:t}) - \xi_{t-1}(X^{\mathbf{k}}_{1:t-1}) H^{\mathbf{k}}_{t-1}], \\ \epsilon^j_{2t} &= K^{-m+1} \sum_{\mathbf{k}\in\mathbb{Z}^m : A^{k(m)}_{m,t-1}=j} (\#^{k(m)}_t - K W^{k(m)}_t) \xi_t(X^{\mathbf{k}}_{1:t}) \tilde{H}^{\mathbf{k}}_t. \end{aligned}$$

Then for each j and m, $\{\epsilon_u^j, \mathcal{F}_u, 2(m-1)T < u \leq 2mT\}$ is a martingale difference sequence, and

$$K(\hat{\psi}_{MT} - \psi_{MT}) = \sum_{m=1}^{M}\sum_{j=1}^{K}(\epsilon^j_{2(m-1)T+1} + \cdots + \epsilon^j_{2mT}). \tag{3.4.1}$$

Therefore, $\mathbb{E}[\hat{\psi}_{MT}] = \psi_{MT}$.

Proof:

The proof of this theorem is similar to that of Lemma 3.3 and Theorem 3.4. We shall give a brief outline of the proof as it can be expanded using similar argument.

Since $\#_t^k$ follows a binomial distribution with parameters K and W_t^k when conditioned on $\mathcal{F}_{2t-1}$, by the tower property of conditional expectations (Theorem 1.1),

$$\begin{aligned}
\mathbb{E}[\epsilon^j_{2t-1}|\mathcal{F}_{2t-2}] &= K^{-m+1}\sum_{\mathbf{k}\in\mathbb{Z}^m:A^{k(m)}_{m,t-1}=j}\{\mathbb{E}[\xi_t(\tilde{X}^{\mathbf{k}}_{1:t})|\mathcal{F}_{2t-2}] - \xi_{t-1}(X^{\mathbf{k}}_{1:t-1})\}H^{\mathbf{k}}_{t-1} = 0,\\
\mathbb{E}[\epsilon^j_{2t}|\mathcal{F}_{2t-1}] &= K^{-m+1}\sum_{\mathbf{k}\in\mathbb{Z}^m:A^{k(m)}_{m,t-1}=j}\{\mathbb{E}[\#_t^{k(m)}|\mathcal{F}_{2t-1}] - KW_t^{k(m)}\}\xi_t(X^{\mathbf{k}}_{1:t})\tilde{H}^{\mathbf{k}}_t = 0.
\end{aligned}$$

Hence, $\{\epsilon_u^j, \mathcal{F}_u, 2(m-1)T < u \leq 2mT\}$ is a martingale difference sequence for each j and m.

Using (3.2.8) and (3.2.9),

$$\begin{aligned}
\sum_{\mathbf{k}\in\mathbb{Z}^m:A^{k(m)}_{m,t-1}=j}(\#_t^{k(m)} - KW_t^{k(m)})\xi_t(X^{\mathbf{k}}_{1:t})\tilde{H}^{\mathbf{k}}_t &= \sum_{\mathbf{k}\in\mathbb{Z}^m:A^{k(m)}_{m,t}=j}\xi_t(X^{\mathbf{k}}_{1:mT})H^{\mathbf{k}}_t\\
&\quad - \sum_{\mathbf{k}\in\mathbb{Z}^m:A^{k(m)}_{m,t-1}=j}\xi_t(\tilde{X}^{\mathbf{k}}_{1:t})H^{\mathbf{k}}_{t-1}.
\end{aligned}$$

By the definition of ϵ^j_{2t-1} and ϵ^j_{2t} and the cancellation of terms using the above equation,

$$\sum_{u=2(m-1)T+1}^{2mT}\epsilon_u^j = K^{-m+1}\left[\sum_{\mathbf{k}\in\mathbb{Z}^m:A^{k(m)}_{m,T}=j}\xi_{mT}(X^{\mathbf{k}}_{1:t})H^{\mathbf{k}}_{mT} - \sum_{\mathbf{k}\in\mathbb{Z}^{m-1}}\xi_{(m-1)T}(X^{\mathbf{k}}_{1:(m-1)T})H^{\mathbf{k}}_{(m-1)T}\right].$$

Accordingly,

$$\sum_{j=1}^{K}\left(\sum_{u=2(m-1)T+1}^{2mT}\epsilon_u^j\right) = \sum_{\mathbf{k}\in\mathbb{Z}^m}\xi_{mT}(X^{\mathbf{k}}_{1:t})\frac{H^{\mathbf{k}}_{mT}}{K^{m-1}} - \sum_{\mathbf{k}\in\mathbb{Z}^{m-1}}\xi_{(m-1)T}(X^{\mathbf{k}}_{1:(m-1)T})\frac{H^{\mathbf{k}}_{(m-1)T}}{K^{m-2}}.$$

From (3.2.6), $\xi_0 = \psi_{MT}$ and $\xi_{MT} = L_c(X_{1:MT})\psi(X_{1:MT})$, using the telescoping property in the above equation, we have

$$\sum_{m=1}^{M}\sum_{j=1}^{K}\left(\sum_{u=2(m-1)T+1}^{2mT}\epsilon_u^j\right) = K^{-1}(\hat{\psi}_{MT} - \psi_{MT}),$$

proving the martingale difference expansion for $\hat{\psi}_{MT}$.

Unbiasedness of $\hat{\psi}_{MT}$ follows immediately due to the martingale difference property for every sequence. □

3.5 Computational time

In running the particle filter for segmented data, one would expect a reduction in the computational cost since we are able to run the multiple particle filters in tandem. For the computational cost of the parallel particle filter, we would need to factor in the computation time involved in the computation of the sum of link weights which is of order $\mathcal{O}(K^2)$ if K particles are used for each particle filter. However, one will be able to reduce the computational cost for the computation of this sum by using different schemes. As an example, the sum could be computed using a recursive algorithm which can be done in $\mathcal{O}(MK)$ operations where M is the number of subsequences used for the parallel particle filter. We will discuss this in detail in Chapter 4, Section 4.5. Another scheme to reduce this computational cost is to make use of sub-sampling, which will be discussed in Chapter 4, Section 4.6. Consequently, the computation of the sum of link weights can be reduced to $\mathcal{O}(K)$ operations. Hence, one would be able to expect significant computational cost savings for our proposed method.

On the note of using multiple processors, suppose we have $N + 1$ processors that we can use for the computation. Let the cost of running a particle filter for MT steps with K particles is C for one processor. Let c_i be the reduction in cost of adding a processor when there are i processors in use. Thus, the cost of the particle filter will

be

$$C - \sum_{i=1}^{N} c_i.$$

In particular, for two processors, the cost will be $C - c_1$. The segmented filter run for $MT/2 \in \mathbb{N}$ steps using K particles will have cost

$$2\left[\frac{C}{2} - \sum_{i=1}^{N/2} c_i\right].$$

where we have assumed $N/2$ is even.

The logic is that one runs two filters (the factor 2) for length $U/2$, accounting for $C/2$ and $2c_i$ and using only half the number of processors. The total computational power used for both methods is the same. The difference of the cost of particle filter minus segmented data particle filter is thus

$$3\sum_{i=1}^{N/2} c_i - \sum_{i=N/2+1}^{N} c_i \tag{3.5.1}$$

which one will expect to be positive since the reduction in cost c_i will be larger for lower indices i. As a result, if the computation time for the sum of link weights is less than (3.5.1), then using parallel particle filter will allow one to gain computational cost savings.

3.6 Choice of proposal density

In practice, we can choose $q_{m,t}(\cdot|x^k_{m,1:t-1}) = f_\theta(\cdot|x^k_{m,t-1})$ to generate the samples at each iteration step. In doing so, the expression for the weights will be simplified and easier to manage. To be specific, consider the m-th subsequence. At time t, we will generate $\tilde{X}^k_{m,1} \sim f_{\theta m}(\cdot)$, $\tilde{X}^k_{m,t} \sim f_\theta(\cdot|x^k_{m,t-1})$ for $1 \le k \le K$ and set $\tilde{X}^k_{m,1:t} = (X^k_{m,1:t-1}, \tilde{X}^k_{m,t})$. We will resample with weights $w_{m,t}(x^k_{m,t}) = g_\theta(Y_{m,t}|x^k_{m,t})$ to obtain $\{X^k_{m,1:t}, 1 \le k \le K\}$ from $\{\tilde{X}^k_{m,1:t}, 1 \le k \le K\}$. However, the simplified algorithm may not be able to achieve optimal performance. As mentioned in Section 2.5, the optimal proposal density is $p_\theta(x_n|x_{n-1}, y_n)$ which is usually intractable in practice. One can seek an approximation of $p_\theta(x_n|x_{n-1}, y_n)$ as the proposal density

to achieve better performance. We shall explore this in Chapter 4 where a numerical study will be performed.

In Chapter 4, we will examine the use of the parallel particle filter to obtain unbiased estimates for the likelihood that will be used in the PMCMC algorithm and substitution algorithm; a numerical study will be conducted. Different options for $f_{\theta m}$ and $q_{m,t}$ will be considered in our simulations as well and will be discussed therein.

Chapter 4

Numerical Study for Likelihood Estimates

4.1 Introduction

In the previous chapter, we have introduced the notation and framework for the parallel particle filter for estimating the likelihood function for a hidden Markov model. We have proven that the likelihood estimator is unbiased in the PPF setting. In this chapter, we will investigate the performance of the estimator by performing simulations on a specific HMM where the parameters are fixed. In Section 4.2, we will discuss the proposed HMM used for our simulation studies and the set of values of parameters chosen. Since the objective of this numerical study is to assess the performance of the likelihood estimate for different setups, the parameter of the HMM will be fixed for the simulation run.

A discussion on the choice of the proposal density used for the particle filters and the initial density for the second subsequence will be discussed in Sections 4.2.2 and 4.2.3 respectively. The results and findings will be reported in Section 4.3. In Section 4.4, we will look into the estimation of smoothed mean via a particle filter (PF) and PPF. In our simulation study, we shall compare the parallel particle filtering with the usual particle filtering (where no segmentation is done to the observed sequence) for different values of the parameter involved. We will further illustrate the recursive computation of the sum of the link weights and thereafter discuss the effect of the

number of subsequences on the computational cost in Section 4.5. We shall see that the estimates obtained from the PPF is comparable to that obtained from the usual particle filtering.

4.2 Proposed HMM

We shall consider the following HMM for this purpose:

$$\begin{aligned} X_{t+1} &= \alpha X_t + \xi_t, \\ Y_t &= X_t + \mu + \nu_t, \end{aligned}$$

for $t = 1, \ldots, MT$. $\xi_t \sim N(0, \sigma_x^2(1-\alpha^2))$ and $\nu_t \sim N(0, \sigma_y^2)$ are independent white noises and $X_1 \sim N(0, \sigma_x^2)$. Since we can set $V_t = Y_t - \mu$, we shall let $\mu = 0$ for our simulation. Under this model, we have

$$\begin{aligned} f_\theta(x_t|x_{t-1}) &= \frac{1}{\sigma_x\sqrt{1-\alpha^2}}\phi\left(\frac{(x_t - \alpha x_{t-1})}{\sigma_x\sqrt{1-\alpha^2}}\right), \\ g_\theta(y_t|x_t) &= \frac{1}{\sigma_y}\phi\left(\frac{y_t - x_t}{\sigma_y}\right), \end{aligned}$$

where $\phi(\cdot)$ is the standard normal density function. Under this setup, the hidden states will have $N(0, \sigma_x^2)$ as the stationary distribution.

We have chosen a linear HMM model for our numerical studies as the exact likelihood can be computed using Kalman filtering. Accordingly, we can assess the performance of the estimator by comparing the exact likelihood with the estimates obtained from the simulations up to ± 1 standard error. For the simulation studies, our objective is to estimate the log likelihood given $2T$ observations. The log likelihood is chosen as it is numerically stable. We shall compare the usual particle filter with the parallel particle filter using $M = 2$ subsequences. The implementation of the PPF using more than two subsequences is similar to that for using two subsequences. The idea is to compute the link weights in a recursive manner and its computational time will be comparable to that for using two subsequences only. We will cover the details in Section 4.5. For the moment, we shall first investigate the use of two subsequences for the parallel particle filtering and examine the time used in the computation of the

particle filters and the sum of link weights.

4.2.1 Selection of parameters' values

For the proposed HMM, there are three parameters (α, σ_x and σ_y) that we can vary and investigate their impact on the log likelihood estimates. Since our proposal densities for both SMC methods depend on α, we shall set $\alpha \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ for our simulations to investigate the impact of α on the performance of the estimator.

Given our choice of the proposal density of $\tilde{X}_{m,t}$ (which shall be discussed later), the performance of particle filtering depends on the significance of the observations $Y_{1:2T}$. We shall consider different values of σ_x/σ_y in our simulations to investigate any observable trends. Accordingly, we shall set $\sigma_x/\sigma_y \in \{\frac{1}{10}, \frac{1}{5}, 1, 5, 10\}$. For each pair of α and σ_x/σ_y, we will compute 10 estimates for the log likelihood to obtain the mean and standard error for the estimate.

We will set $T \in \{50, 100, 150\}$ to examine the effect of T on the method as well for all pairs of parameters' values.

We will use $K = 500$ particles for the particle filter and $K_p = 500$ and $K_p = 700$ particles for the parallel particle filter to compare the computational time involved for different values of K_p. One will hope that the cost of using more particles for the parallel particle filtering will be comparable to that of the usual particle filtering using less particles. If this is true, there will be considerable advantages of using the parallel particle filter as more particles are used without further increasing the computational cost.

4.2.2 The choice of proposal densities

As mentioned in Chapter 3, we will use $q_{m,t}(\cdot|x^k_{m,1:t-1}) = f_\theta(\cdot|x^k_{m,t-1})$ for simplicity. However, for this particular HMM (which is linear), we are able to use the optimal

proposal density using Kalman filtering. Recall from Section 2.5.2 that the optimal proposal density is $p_\theta(x_t|x_{t-1}, y_t)$. Using the Kalman filter, we have

$$q_{opt}(x_t|x_{t-1}, y_t) = \frac{1}{\sqrt{\Sigma_t}}\phi\left(\frac{x_t - m_t}{\sqrt{\Sigma_t}}\right),$$

where

$$\begin{aligned}\Sigma_t^{-1} &= \frac{1}{\sigma_x^2(1-\alpha^2)} + \frac{1}{\sigma_y^2},\\ m_t &= \Sigma_k\left(\frac{\alpha x_{t-1}}{\sigma_x^2(1-\alpha^2)} + \frac{y_t}{\sigma_y^2}\right).\end{aligned}$$

We will compute the likelihood estimate using the usual particle filter for these two different proposal densities. We shall also compare the parallel particle filter using f_θ as the proposal density and compared with the usual particle filter using q_{opt} as the proposal density to compare the performance. We will see that the estimates obtained from the parallel particle filter perform reasonably well even when we compared them with that obtained from the usual particle filter using an optimal proposal density.

4.2.3 Choice of initial distribution for second subsequence

As discussed in the previous chapter, we need to choose a density function, $f_{\theta 2}$, for the density function of $X_{T+1} := X_{2,1}$ in the second subsequence. Referring to (3.2.1), for the case when $M = 2$, the estimate of the likelihood function can be simplified to

$$\hat{\eta}(\theta) = \left[\prod_{m=1}^{2}\prod_{t=1}^{T}\bar{w}_{m,t}\right]\left[\frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K}\frac{f_\theta(x_{2,1}^j|x_T^k)}{f_{\theta 2}(x_{2,1}^j)}\right]. \tag{4.2.1}$$

The expression (4.2.1) suggests a good choice for $f_{\theta 2}$. Consider the SMC approximation of $f_\theta(\cdot|x_T)$ given by

$$\hat{f}_\theta(x) := \frac{1}{K}\sum_{k=1}^{K} f_\theta(x|x_T^k).$$

If we set $f_{\theta 2}(x) = \hat{f}_\theta(x)$, then the double sum in (4.2.1) will sum to K^2. Consequently, with this choice of $f_{\theta 2}$, the double sum will not contribute to the variance of the estimate. This indicates that $\hat{f}_\theta$ will be an optimal choice of $f_{\theta 2}$. However, to obtain $\hat{f}_\theta$, one has to complete the first particle filter prior to the execution of the second

particle filter. Thus, this choice of $f_{\theta 2}$ will not be reasonable as it contradicts the objective of using parallel computation. With this result in mind, we should choose $f_{\theta 2}$ such that it will approximate $f_\theta(\cdot|x_T)$.

Another possible choice of $f_{\theta 2}$ is the density function of X_{T+1} given $Y_{1:T}$, f_{T+1}. However, this density function is hard to obtain in practice. For a better performance for the SMC method, one would consider a choice of $f_{\theta 2}$ as close as possible to f_{T+1}. Accordingly, if we are able to obtain a good approximation of the actual distribution, one would expect the performance of the estimate to be comparable to that when we use the actual distribution of X_{T+1} for $f_{\theta 2}$. We would like the performance of this estimate to be comparable to the usual particle filter as well and having a good approximation of f_{T+1} may provide us with such an estimate. Thus, in our simulation studies, we would like to investigate how different choice of $f_{\theta 2}$ will affect the performance of the parallel particle filter estimates.

For our choice of the HMM, we will be able to compute the actual distribution of X_{T+1} given $Y_{1:T}$ using Kalman filtering. However, in practice, one will not be able to obtain the actual distribution. Accordingly, we should consider an approximation of the distribution of X_{T+1} given $Y_{1:T}$ for our choice of $f_{\theta 2}$ in general. We would also consider a distribution that provides a bad approximation to f_{T+1} to examine how it will affect the performance of the log likelihood estimate.

Summing up, for our simulation studies, we shall consider the following options:

(i) $X_{T+1} \sim N(0, \sigma_x^2)$,

(ii) $X_{T+1} \sim p(\cdot|Y_{1:T})$, the density for X_{T+1} given $Y_{1:T}$,

(iii) $X_{T+1} \sim \tilde{p}(\cdot|Y_{1:T})$ where $\tilde{p}$ is an approximation of the density $p(\cdot|Y_{1:T})$.

For (ii), we will use Kalman filtering to obtain the mean and variance of the normal density function. For (iii), we will make use of particle filtering to obtain $\tilde{p}$. We shall now give details for these two approaches.

Referring to Algorithm 1, we have the following pseudo code to obtain the mean and variance of $p(\cdot|Y_{1:T})$ in Algorithm 13. With this, for the second subsequence, we will generate $\tilde{X}_{T+1} \sim N(\hat{\mu}_{T+1|T}, \Sigma_{T+1|T})$ and perform the usual particle filter.

Algorithm 13: Kalman Filtering to obtain mean and variance of $p(\cdot|Y_{1:T})$

begin

Initialisation;

$K_1 = \frac{\sigma_x^2}{\sigma_x^2+\sigma_y^2}$, $\hat{\mu}_{1|1} = \frac{\sigma_x^2 Y_1}{\sigma_x^2+\sigma_y^2}$, $\Sigma_{1|1} = \frac{\sigma_x^2\sigma_y^2}{\sigma_x^2+\sigma_y^2}$;

for $k = 2 : T$ **do**

Compute the following:

$$\hat{\mu}_{k|k-1} = \alpha\hat{\mu}_{k-1|k-1}, \qquad \text{Prediction}$$

$$\Sigma_{k|k-1} = \alpha^2\Sigma_{k-1|k-1} + \sigma_x^2(1-\alpha^2), \qquad \text{Prediction}$$

$$K_k = \frac{\Sigma_{k|k-1}}{\Sigma_{k|k-1} + \sigma_y^2}, \qquad \text{Kalman gain}$$

$$\hat{\mu}_{k|k} = (1-K_k)\hat{\mu}_{k|k-1} + K_k Y_k, \qquad \text{filter state estimation}$$

$$\Sigma_{k|k} = (1-K_k)\Sigma_{k|k-1}, \qquad \text{filter error covariance.}$$

end

end

For (iii), the idea is to obtain an approximation by considering $r+1$ observations prior to X_{T+1}. That is, we will perform particle filtering based on $Y_{T-r:T}$ and using $X_{T-r} \sim N(0, \sigma_x^2)$. The particles obtained will then be used to compute an estimate for the mean and variance of the density function. The pseudo code of this algorithm is given in Algorithm 14. With this, for the second subsequence, we will generate $\tilde{X}_{T+1} \sim N(\hat{\mu}_T, \hat{\sigma}_T^2)$ and perform the usual particle filter.

Algorithm 14: Particle Filtering to obtain mean and variance of $\tilde{p}(\cdot|Y_{1:T})$

begin

  for $k = 1 : K$ **do**

    Choose r and simulate $\tilde{X}^k_{T-r} \sim N(0, \sigma_x^2)$ and compute $w^k_{T-r} = g_\theta(Y_{T-r}|\tilde{X}^k_{T-r})$, then perform multinomial resampling to obtain X^k_{T-r};

  end

  for $t = T - r + 1 : T$ **do**

    for $k = 1 : K$ **do**

      Generate $\tilde{X}^k_t \sim f_\theta(\cdot|X^k_{t-1})$, compute $w^k_t = g_\theta(Y_t|\tilde{X}^k_t)$ and perform multinomial resampling to obtain X^k_t;

    end

  end

  Compute $\hat{\mu}_T := \frac{1}{K}\sum_{k=1}^K X^k_T$, $\hat{\sigma}^2_T := \frac{1}{K-1}\sum_{k=1}^K (X^k_T - \hat{\mu}_T)^2$

end

4.3 Numerical results

Throughout this section, we will denote l_T to be the log likelihood obtained using Kalman filtering. We will use the following notations for the estimates obtained

(i) $\hat{l}$ is the log likelihood estimate obtained using the usual particle filtering using f_θ,

(ii) $\hat{l}_{opt}$ is the log likelihood estimate obtained using the usual particle filtering using q_{opt},

(iii) $\tilde{l}_1$ is the log likelihood estimate obtained using parallel SMC method with $X_{T+1} \sim N(0, \sigma_x^2)$,

(iv) $\tilde{l}_2$ is the log likelihood estimate obtained using parallel SMC method with $X_{T+1} \sim p(\cdot|Y_{1:T})$,

(v) $\tilde{l}_3$ is the log likelihood estimate obtained using parallel SMC method with $X_{T+1} \sim \tilde{p}(\cdot|Y_{1:T})$.

For all the parallel SMC methods, we have set $M = 2$ and run the program such that the second subsequence is computed after the first subsequence. Thereafter, the link weights will be computed for the log likelihood estimate. For all estimates, we will report the mean estimate and its standard error. The computational time will be reported as well.

4.3.1 Tables of simulation results

We will first present the simulation results obtained for different setups in Tables 4.1 to 4.18. The discussion of these results will be done in Section 4.3.2, Section 4.3.3 and Section 4.3.4.

	$T = 50$	$T = 100$	$T = 150$
$\hat{l}$	-144.61 ± 0.1561	-287.84 ± 0.20	-427.82 ± 0.54
$\hat{l}_{opt}$	-144.39 ± 0.04	-287.75 ± 0.04	-427.35 ± 0.03
$\tilde{l}_1$	-145.10 ± 0.33	-288.51 ± 0.35	-428.81 ± 0.25
$\tilde{l}_2$	-144.77 ± 0.20	-288.26 ± 0.28	-427.77 ± 0.57
$\tilde{l}_3$	-144.27 ± 0.28	-288.80 ± 0.35	-428.54 ± 0.52
l_T	-144.32	-287.70	-427.41

Table 4.1: Comparison of Log Likelihood Estimates for Different Values of T with $\alpha = 0.9$, $\sigma_x/\sigma_y = 10$, $K = K_p = 500$.

	$T = 50$	$T = 100$	$T = 150$
$\hat{l}$	-164.96 ± 0.27	-328.69 ± 0.73	-480.94 ± 0.66
$\hat{l}_{opt}$	-164.69 ± 0.02	-326.97 ± 0.02	-480.07 ± 0.05
$\tilde{l}_1$	-165.55 ± 0.34	-327.70 ± 0.53	-480.38 ± 0.52
$\tilde{l}_2$	-164.91 ± 0.35	-328.06 ± 0.63	-481.87 ± 0.48
$\tilde{l}_3$	-164.93 ± 0.22	-327.92 ± 0.72	-481.86 ± 0.36
l_T	-164.66	-326.99	-480.07

Table 4.2: Comparison of Log Likelihood Estimates for Different Values of T with $\alpha = 0.8$, $\sigma_x/\sigma_y = 10$, $K = K_p = 500$.

	$T = 50$	$T = 100$	$T = 150$
$\hat{l}$	-170.69 ± 0.30	-342.79 ± 0.53	-504.89 ± 0.56
$\hat{l}_{opt}$	-170.27 ± 0.02	-339.99 ± 0.02	-503.23 ± 0.02
$\tilde{l}_1$	-170.04 ± 0.31	-342.38 ± 0.47	-505.03 ± 0.60
$\tilde{l}_2$	-170.88 ± 0.36	-342.09 ± 0.53	-505.62 ± 0.62
$\tilde{l}_3$	-171.30 ± 0.34	-341.61 ± 0.51	-503.92 ± 0.51
l_T	-170.25	-340.01	-503.24

Table 4.3: Comparison of Log Likelihood Estimates for Different Values of T with $\alpha = 0.7$, $\sigma_x/\sigma_y = 10$, $K = K_p = 500$.

	$T = 50$	$T = 100$	$T = 150$
$\hat{l}$	-172.58 ± 0.31	-344.35 ± 0.60	-521.67 ± 0.63
$\hat{l}_{opt}$	-172.27 ± 0.01	-343.23 ± 0.01	-519.64 ± 0.02
$\tilde{l}_1$	-172.85 ± 0.38	-344.12 ± 0.66	-521.18 ± 0.62
$\tilde{l}_2$	-173.08 ± 0.53	-344.63 ± 0.41	-521.11 ± 0.71
$\tilde{l}_3$	-173.32 ± 0.38	-345.22 ± 0.25	-520.10 ± 0.50
l_T	-172.26	-343.21	-519.65

Table 4.4: Comparison of Log Likelihood Estimates for Different Values of T with $\alpha = 0.6$, $\sigma_x/\sigma_y = 10$, $K = K_p = 500$.

	$T = 50$	$T = 100$	$T = 150$
$\hat{l}$	-183.16 ± 0.35	-359.35 ± 0.55	-539.90 ± 0.62
$\hat{l}_{opt}$	-183.25 ± 0.01	-358.36 ± 0.01	-538.29 ± 0.02
$\tilde{l}_1$	-184.64 ± 0.73	-359.54 ± 0.52	-540.43 ± 0.73
$\tilde{l}_2$	-184.09 ± 0.54	-360.46 ± 0.38	-539.89 ± 0.71
$\tilde{l}_3$	-185.03 ± 0.50	-359.54 ± 0.47	-540.47 ± 0.68
l_T	-183.24	-358.36	-538.28

Table 4.5: Comparison of Log Likelihood Estimates for Different Values of T with $\alpha = 0.5$, $\sigma_x/\sigma_y = 10$, $K = K_p = 500$.

	$T = 50$	$T = 100$	$T = 150$
$\hat{l}$	-155.71 ± 0.31	-302.03 ± 0.38	-449.39 ± 0.32
$\hat{l}_{opt}$	-155.60 ± 0.03	-301.54 ± 0.05	-449.53 ± 0.06
$\tilde{l}_1$	-156.15 ± 0.17	-302.06 ± 0.36	-450.38 ± 0.53
$\tilde{l}_2$	-154.89 ± 0.29	-302.15 ± 0.30	-450.83 ± 0.49
$\tilde{l}_3$	-155.87 ± 0.26	-301.66 ± 0.33	-449.71 ± 0.59
l_T	-155.57	-301.52	-449.46

Table 4.6: Comparison of Log Likelihood Estimates for Different Values of T with $\alpha = 0.9$, $\sigma_x/\sigma_y = 10$, $K = 500$, $K_p = 700$.

	$T = 50$	$T = 100$	$T = 150$
$\hat{l}$	-156.08 ± 0.22	-319.55 ± 0.40	-477.56 ± 0.63
$\hat{l}_{opt}$	-155.97 ± 0.01	-318.01 ± 0.03	-476.70 ± 0.03
$\tilde{l}_1$	-156.57 ± 0.27	-318.64 ± 0.40	-477.93 ± 0.48
$\tilde{l}_2$	-156.34 ± 0.24	-317.72 ± 0.34	-477.26 ± 0.42
$\tilde{l}_3$	-156.23 ± 0.25	-317.96 ± 0.52	-477.59 ± 0.53
l_T	-156.00	-318.02	-476.65

Table 4.7: Comparison of Log Likelihood Estimates for Different Values of T with $\alpha = 0.8$, $\sigma_x/\sigma_y = 10$, $K = 500$, $K_p = 700$.

	$T = 50$	$T = 100$	$T = 150$
$\hat{l}$	-172.20 ± 0.53	-344.77 ± 0.56	-514.79 ± 0.55
$\hat{l}_{opt}$	-171.04 ± 0.01	-343.48 ± 0.01	-513.36 ± 0.02
$\tilde{l}_1$	-171.85 ± 0.16	-344.68 ± 0.53	-514.41 ± 0.59
$\tilde{l}_2$	-171.24 ± 0.21	-345.28 ± 0.40	-514.60 ± 0.80
$\tilde{l}_3$	-171.81 ± 0.28	-344.35 ± 0.53	-515.99 ± 0.89
l_T	-171.07	-343.44	-513.36

Table 4.8: Comparison of Log Likelihood Estimates for Different Values of T with $\alpha = 0.7$, $\sigma_x/\sigma_y = 10$, $K = 500$, $K_p = 700$.

	$T = 50$	$T = 100$	$T = 150$
$\hat{l}$	-176.24 ± 0.29	-356.02 ± 0.56	-524.65 ± 0.60
$\hat{l}_{opt}$	-176.29 ± 0.01	-353.89 ± 0.02	-523.02 ± 0.02
$\tilde{l}_1$	-177.22 ± 0.34	-354.38 ± 0.25	-523.26 ± 0.38
$\tilde{l}_2$	-176.79 ± 0.32	-354.27 ± 0.29	-523.80 ± 0.55
$\tilde{l}_3$	-176.29 ± 0.40	-354.83 ± 0.50	-524.17 ± 0.39
l_T	-176.29	-353.88	-523.05

Table 4.9: Comparison of Log Likelihood Estimates for Different Values of T with $\alpha = 0.6$, $\sigma_x/\sigma_y = 10$, $K = 500$, $K_p = 700$.

	$T = 50$	$T = 100$	$T = 150$
$\hat{l}$	-180.66 ± 0.55	-366.62 ± 0.39	-545.47 ± 0.56
$\hat{l}_{opt}$	-180.46 ± 0.01	-364.69 ± 0.02	-543.47 ± 0.02
$\tilde{l}_1$	-181.10 ± 0.28	-366.02 ± 0.42	-545.02 ± 0.48
$\tilde{l}_2$	-180.94 ± 0.29	-366.18 ± 0.44	-546.09 ± 1.16
$\tilde{l}_3$	-180.87 ± 0.33	-366.03 ± 0.47	-546.06 ± 0.69
l_T	-180.48	-364.68	-543.45

Table 4.10: Comparison of Log Likelihood Estimates for Different Values of T with $\alpha = 0.5$, $\sigma_x/\sigma_y = 10$, $K = 500$, $K_p = 700$.

	Computation Time			
$\hat{l}$	0.3016			
$\hat{l}_{opt}$	1.0123			
	$K_p = 500$		$K_p = 700$	
	T_f	T_l	T_f	T_l
$\tilde{l}_1$	0.1507	0.0446	0.2013	0.0779
$\tilde{l}_2$	0.1523	0.0421	0.2039	0.0781
$\tilde{l}_3$	0.1471	0.0403	0.2015	0.0767

Table 4.11: Comparison of Computational Time for $T = 50$.

	Computation Time			
$\hat{l}$	0.5499			
$\hat{l}_{opt}$	1.7359			
	$K_p = 500$		$K_p = 700$	
	T_f	T_l	T_f	T_l
$\tilde{l}_1$	0.3119	0.0473	0.3702	0.0884
$\tilde{l}_2$	0.3102	0.0473	0.3743	0.0783
$\tilde{l}_3$	0.3143	0.0508	0.3756	0.0790

Table 4.12: Comparison of Computational Time for $T = 100$.

	Computation Time			
$\hat{l}$	0.8321			
$\hat{l}_{opt}$	2.7738			
	$K_p = 500$		$K_p = 700$	
	T_f	T_l	T_f	T_l
$\tilde{l}_1$	0.4693	0.0444	0.5639	0.0768
$\tilde{l}_2$	0.4859	0.0456	0.5604	0.0764
$\tilde{l}_3$	0.4704	0.0481	0.6544	0.0863

Table 4.13: Comparison of Computational Time for $T = 150$.

	$\alpha = 0.9$	$\alpha = 0.8$	$\alpha = 0.7$	$\alpha = 0.6$	$\alpha = 0.5$
$\hat{l}$	-144.61 ± 0.16	-164.96 ± 0.27	-170.69 ± 0.30	-172.58 ± 0.31	-183.16 ± 0.35
$\hat{l}_{opt}$	-144.39 ± 0.04	-164.69 ± 0.02	-170.27 ± 0.02	-172.27 ± 0.01	-183.25 ± 0.01
$\tilde{l}_1$	-145.10 ± 0.33	-165.56 ± 0.34	-170.04 ± 0.31	-172.85 ± 0.38	-184.64 ± 0.73
$\tilde{l}_2$	-144.78 ± 0.20	-164.91 ± 0.35	-170.88 ± 0.36	-173.08 ± 0.53	-184.09 ± 0.54
$\tilde{l}_3$	-144.27 ± 0.28	-164.93 ± 0.22	-171.30 ± 0.34	-173.32 ± 0.38	-185.03 ± 0.50
l_T	-144.32	-164.66	-170.25	-172.26	-183.24

Table 4.14: Comparison of Log Likelihood Estimates for Different Values of α with $T = 50$, $\sigma_x/\sigma_y = 10$, $K = K_p = 500$.

	$\alpha = 0.9$	$\alpha = 0.8$	$\alpha = 0.7$	$\alpha = 0.6$	$\alpha = 0.5$
$\hat{l}$	-119.35 ± 0.19	-126.25 ± 0.18	-129.49 ± 0.25	-148.55 ± 0.46	-143.15 ± 0.25
$\hat{l}_{opt}$	-118.99 ± 0.04	-126.00 ± 0.03	-129.29 ± 0.03	-147.86 ± 0.02	-142.79 ± 0.02
$\tilde{l}_1$	-119.09 ± 0.16	-126.45 ± 0.17	-129.57 ± 0.22	-148.55 ± 0.28	-143.13 ± 0.29
$\tilde{l}_2$	-119.29 ± 0.11	-126.43 ± 0.14	-129.38 ± 0.17	-148.40 ± 0.36	-142.90 ± 0.29
$\tilde{l}_3$	-120.89 ± 0.37	-127.09 ± 0.47	-130.12 ± 0.31	-151.66 ± 0.35	-142.97 ± 0.17
l_T	-118.95	-126.04	-129.32	-147.86	-142.77

Table 4.15: Comparison of Log Likelihood Estimates for Different Values of α with $T = 50$, $\sigma_x/\sigma_y = 5$, $K = K_p = 500$.

	$\alpha = 0.9$	$\alpha = 0.8$	$\alpha = 0.7$	$\alpha = 0.6$	$\alpha = 0.5$
$\hat{l}$	-83.28 ± 0.07	-80.35 ± 0.10	-89.33 ± 0.12	-86.32 ± 0.08	-88.20 ± 0.07
$\hat{l}_{opt}$	-83.41 ± 0.08	-80.30 ± 0.05	-89.22 ± 0.08	-86.12 ± 0.06	-88.16 ± 0.05
$\tilde{l}_1$	-83.65 ± 0.18	-80.41 ± 0.08	-89.28 ± 0.11	-86.08 ± 0.08	-88.35 ± 0.16
$\tilde{l}_2$	-83.37 ± 0.07	-80.22 ± 0.08	-89.13 ± 0.11	-86.10 ± 0.07	-88.33 ± 0.10
$\tilde{l}_3$	-83.41 ± 0.05	-80.47 ± 0.11	-89.18 ± 0.06	-86.28 ± 0.06	-88.46 ± 0.08
l_T	-83.24	-80.31	-89.19	-86.12	-88.21

Table 4.16: Comparison of Log Likelihood Estimates for Different Values of α with $T = 50$, $\sigma_x/\sigma_y = 1$, $K = K_p = 500$.

	$\alpha = 0.9$	$\alpha = 0.8$	$\alpha = 0.7$	$\alpha = 0.6$	$\alpha = 0.5$
$\hat{l}$	-162.09 ± 0.02	-145.89 ± 0.02	-156.05 ± 0.03	-154.49 ± 0.02	-159.76 ± 0.03
$\hat{l}_{opt}$	-162.16 ± 0.06	-145.98 ± 0.05	-156.02 ± 0.04	-154.52 ± 0.02	-159.76 ± 0.02
$\tilde{l}_1$	-162.13 ± 0.04	-145.99 ± 0.04	-156.07 ± 0.04	-154.51 ± 0.02	-159.78 ± 0.03
$\tilde{l}_2$	-162.11 ± 0.04	-145.95 ± 0.03	-156.07 ± 0.03	-154.50 ± 0.02	-159.77 ± 0.03
$\tilde{l}_3$	-162.09 ± 0.05	-145.95 ± 0.03	-156.03 ± 0.03	-154.54 ± 0.05	-159.81 ± 0.10
l_T	-162.11	-145.95	-156.05	-154.51	-159.76

Table 4.17: Comparison of Log Likelihood Estimates for Different Values of α with $T = 50$, $\sigma_x/\sigma_y = \frac{1}{5}$, $K = K_p = 500$.

	$\alpha = 0.9$	$\alpha = 0.8$	$\alpha = 0.7$	$\alpha = 0.6$	$\alpha = 0.5$
$\hat{l}$	-188.39 ± 0.01	-192.21 ± 0.03	-188.88 ± 0.01	-186.97 ± 0.01	-177.64 ± 0.01
$\hat{l}_{opt}$	-188.31 ± 0.09	-192.18 ± 0.04	-188.88 ± 0.02	-186.99 ± 0.01	-177.66 ± 0.01
$\tilde{l}_1$	-188.39 ± 0.02	-192.22 ± 0.04	-188.94 ± 0.02	-187.01 ± 0.02	-177.64 ± 0.01
$\tilde{l}_2$	-188.37 ± 0.03	-192.20 ± 0.03	-188.88 ± 0.02	-187.01 ± 0.02	-177.66 ± 0.01
$\tilde{l}_3$	-188.39 ± 0.05	-192.19 ± 0.03	-188.87 ± 0.05	-187.02 ± 0.04	-177.66 ± 0.05
l_T	-188.38	-192.20	-188.88	-186.99	-177.65

Table 4.18: Comparison of Log Likelihood Estimates for Different Values of α with $T = 50$, $\sigma_x/\sigma_y = \frac{1}{10}$, $K = K_p = 500$.

4.3.2 Comparison for different values of T

We shall first examine the values of the estimates for different values of T. The results for the simulations are summarised in Tables 4.1 to 4.10 for different values of σ_x/σ_y. Table 4.11 to 4.13 give the computation time (in seconds) for the particles filters under different setup.

Simulation time

We shall refer to Table 4.11 to 4.13 for the comparison of simulation time. We shall denote T_f to be the mean computation time for the particle filter and T_l to be the mean computation time for the sum of link weights. We remark that the computation of the sum of link weights can be performed using vector term-by-term multiplication

which results in computational cost savings. This will be elaborated in Section 4.5.

For the usual particle filter, the simulation time for $T = 100$ is around 1.8 times that for $T = 50$. Similarly, the simulation time for $T = 150$ is around 2.8 times that for $T = 50$. This is expected as the computation time is of $\mathcal{O}(T)$.

The time taken for the usual particle filter using the optimal proposal density is more than thrice that of the usual particle filter using f_θ as the proposal density. The additional time is due to the computation of the mean and standard deviation of the proposal density at each value of T for the simulation of the random variables involved.

From Table 4.11, if the segmented particle filters are run in parallel, the combined time (factoring in the computation of the sum of link weights) for $K_p = 500$ is less than the time taken for the usual particle filter using the same number of particles. The cost saving is around one third of the particle filter computational time. If $K_p = 700$, the combined time is still less than the time taken for the usual particle filter using $K = 500$ particles. The cost saving is around one tenth of the particle filter computational time. Accordingly, for the parallel particle filter, one can use more particles for the computation of the estimates and at the same time obtain computational cost savings. Similar conclusions can be made for the case when $T = 100$ and $T = 150$ by referring to Table 4.12 and Table 4.13. This gives an advantage for the parallel particle filter over the usual particle filter by gaining computational cost savings while using more particles to compute the estimates.

Performance of estimates

We shall first examine the performance of the estimates for $K = K_p = 500$. From Table 4.1 to 4.5, we can see that for the parallel particle filter using option (i), the exact log likelihood does not lie in the interval obtain from the computed estimates for most cases. This could be due to the bad choice of $f_{\theta 2}$ used in the algorithm. However, the estimates obtained are not too far off from the exact log likelihood, suggesting that the estimates obtained with a poor choice of $f_{\theta 2}$ would be able to give

a reasonable estimate for the log likelihood.

For the parallel particle filter using option (ii) and option (iii), we have similar conclusion as above. However, for most cases, the standard errors obtained are smaller compared to that using parallel particle filter using option (i). This suggests that the estimates obtain using option (ii) or option (iii) are less variable and more stable. Further, comparing the estimates using the usual particle filter with f_θ as the proposal density, the parallel particle filter using option (ii) and option (iii), we see that the standard errors of the estimates are comparable. Thus, in using the parallel particle filter with appropriate $f_{\theta 2}$, we can obtain estimates that have comparable performance to the usual particle filter using the same proposal density and able to save computational time. If the same proposal density is used for the usual particle filter and parallel particle filter, one would expect comparable performance between these two methods. This is an encouraging result for using parallel particle filter to obtain log likelihood estimates.

When we have $K = 500$ and $K_p = 700$, one would expect the parallel particle filter to perform better as more particles are used in the computation of the log likelihood estimate. From Table 4.6 to 4.10, this is true for most cases. The standard errors for the estimates obtained using option (ii) and (iii) are usually smaller than the estimates computed using the usual particle filter. This suggests that the parallel particle filter estimates are stable for most cases. In using more particles, one can achieve better estimates and at the same time gain computational cost savings. This would make our proposed method more attractive than the usual particle filter.

4.3.3 Comparison for different values of α

We shall consider the effect of α for $T = 50$ and different values of σ_x/σ_y. Note that the signal to noise ratio (SNR) is $\sigma_x^2(1-\alpha^2)/\sigma_y^2$. Accordingly, for a fixed value of σ_x/σ_y, the SNR will increase as α decreases.

We shall refer to Table 4.14 to 4.18 for our analysis. When $\sigma_x/\sigma_y = 10$, the standard error of the estimates obtain from the usual particle filter and parallel particle filter using option (i) to (iii) increases as α decreases. However, this trend is not evident for the other values of σ_x/σ_y. For high SNR, one may need a better proposal density to address this problem.

For cases with low SNR (Table 4.16 to 4.18), the parallel particle filter using option (ii) and (iii) are comparable to the usual particle filter for the five different values of α chosen. Under these settings, the choice of f_θ as the proposal density works well and choosing an appropriate $f_{\theta 2}$ will improve the performance of the estimates (using standard errors as a basis for comparison).

4.3.4 Comparison for different values of σ_x/σ_y

We shall consider the effect of σ_x/σ_y for $T = 50$ and setting $\alpha = 0.7$. One would expect the particle filters to perform better when the SNR is low. The results are summarised in Table 4.19 for easy reference.

	$\sigma_x/\sigma_y = 10$	$\sigma_x/\sigma_y = 5$	$\sigma_x/\sigma_y = 1$	$\sigma_x/\sigma_y = \frac{1}{5}$	$\sigma_x/\sigma_y = \frac{1}{10}$
$\hat{l}$	-170.69 ± 0.30	-129.49 ± 0.25	-89.33 ± 0.12	-156.05 ± 0.03	-188.88 ± 0.01
$\hat{l}_{opt}$	-170.27 ± 0.02	-129.29 ± 0.03	-89.22 ± 0.08	-156.02 ± 0.04	-188.88 ± 0.02
$\tilde{l}_1$	-170.04 ± 0.31	-129.57 ± 0.22	-89.28 ± 0.11	-156.07 ± 0.04	-188.94 ± 0.02
$\tilde{l}_2$	-170.88 ± 0.36	-129.38 ± 0.17	-89.13 ± 0.11	-156.07 ± 0.03	-188.88 ± 0.02
$\tilde{l}_3$	-171.30 ± 0.34	-130.12 ± 0.31	-89.18 ± 0.06	-156.03 ± 0.03	-188.87 ± 0.05
l_T	-170.25	-129.32	-89.19	-156.05	-188.88

Table 4.19: Comparison of Log Likelihood Estimates for Different Values of σ_x/σ_y with $T = 50$, $\alpha = 0.7$, $K = K_p = 500$.

From the above table, we can observe that the standard error of all the estimates decreases as σ_x/σ_y decreases. This is expected as the SNR decreases as the ratio decreases. The performance of the parallel particle filter estimates are comparable to the usual particle filter for all the three options of $f_{\theta 2}$ chosen for the five different settings.

4.4 Estimation of smoothed means

In this section, we will examine the performance of parallel particle filter in the estimation of smoothed means. We will use 2 subsequences as before. Due to repeated resampling, for the usual particle filter, we would expect less distinct values of X_u where $u \leq T$ for the particles as compared to that obtained from the parallel particle filter. We will perform simulations to ascertain this.

Recall from equation (3.2.6) that an estimator of $\psi_{2T} := \mathbb{E}_p[\psi(X_{1:2T})|Y_{1:2T}]$ using the parallel particle filter is given by

$$\tilde{\psi}_{2T} = \frac{\sum_{k=1}^{K}\sum_{l=1}^{K} L(X_{1:2T}^{kl})\psi(X_{1:2T}^{kl})H_T^{kl}}{\sum_{k=1}^{K}\sum_{l=1}^{K} L(X_{1:2T}^{kl})H_T^{kl}} \tag{4.4.1}$$

where $H_{1:T}^{kl} = H_{1,T}^{k}H_{2,T}^{l}$ and $X_{1:2T}^{kl} = (X_{1,1:T}^{k}, X_{2,1:T}^{l})$.

For the usual particle filter, the expression will simplify to

$$\tilde{\psi}_{2T} = \frac{1}{K}\sum_{k=1}^{K}\psi(X_{1:2T}^{k}).$$

As a comparison, we will consider $\psi_{2T} = X_{10}$ and $\psi_{2T} = X_{40}$ for the case when $T = 50$, $\alpha = 0.7$ and different values of σ_x/σ_y. We will compute the estimates using the above estimate given in equation (4.4.1). We will use the same number of particles for the usual particle filter and parallel particle filter. We have also included the usual particle filter with optimal proposal density as a basis for comparison. The results are summarised in Table 4.20 to 4.24.

For high SNR (Table 4.20 and Table 4.21), the parallel particle filter smoothed means estimates has lower standard error for $\tilde{X}_{10}$ compared to the usual particle filter. For $\sigma_x = \sigma_y$, the performance of the parallel SMC with the usual particle filter is comparable. The behaviour of the estimates for low SNR tends to be erratic, which can be seen from Table 4.23 and Table 4.24. To explain this, we shall look at the hidden states trajectory for the particle filter using optimal proposal density to get a better idea. Figure 4.1 to Figure 4.5 give the hidden states trajectory for different SNR values.

	$\tilde{X}_{10}$	$\tilde{X}_{40}$
Usual PF with Optimal Proposal	1.6733 ± 0.1528	-20.3140 ± 0.0859
Usual PF	1.6188 ± 0.3582	-20.2353 ± 0.2016
Parallel SMC (i)	1.5659 ± 0.1865	-20.2346 ± 0.0722
Parallel SMC (ii)	1.7391 ± 0.2177	-20.3657 ± 0.1390
Parallel SMC (iii)	1.5418 ± 0.1809	-20.2599 ± 0.1311

Table 4.20: Comparison of Hidden States Estimates for $\sigma_x/\sigma_y = 10$.

	$\tilde{X}_{10}$	$\tilde{X}_{40}$
Usual PF with Optimal Proposal	-7.2638 ± 0.1751	-2.4171 ± 0.1093
Usual PF	-6.9270 ± 0.3238	-2.3133 ± 0.2242
Parallel SMC (i)	-7.4086 ± 0.1436	-2.2331 ± 0.1527
Parallel SMC (ii)	-7.1260 ± 0.1262	-2.3728 ± 0.1278
Parallel SMC (iii)	-7.5336 ± 0.1931	-2.1327 ± 0.2401

Table 4.21: Comparison of Hidden States Estimates for $\sigma_x/\sigma_y = 5$.

	$\tilde{X}_{10}$	$\tilde{X}_{40}$
Usual PF with Optimal Proposal	-0.6899 ± 0.0860	0.7732 ± 0.0525
Usual PF	-0.8459 ± 0.0774	0.7054 ± 0.0287
Parallel SMC (i)	-0.6608 ± 0.1130	0.7815 ± 0.0577
Parallel SMC (ii)	-0.5873 ± 0.0816	0.6312 ± 0.0555
Parallel SMC (iii)	-0.7458 ± 0.0568	0.7727 ± 0.0681

Table 4.22: Comparison of Hidden States Estimates for $\sigma_x/\sigma_y = 1$.

	$\tilde{X}_{10}$	$\tilde{X}_{40}$
Usual PF with Optimal Proposal	-0.4814 ± 0.1883	-0.0661 ± 0.1111
Usual PF	-0.1207 ± 0.1312	0.0649 ± 0.1064
Parallel SMC (i)	-0.3524 ± 0.0983	-0.0963 ± 0.0912
Parallel SMC (ii)	-0.5552 ± 0.1361	0.0376 ± 0.0818
Parallel SMC (iii)	-0.2817 ± 0.1228	-0.0653 ± 0.0647

Table 4.23: Comparison of Hidden States Estimates for $\sigma_x/\sigma_y = \frac{1}{5}$.

	$\tilde{X}_{10}$	$\tilde{X}_{40}$
Usual PF with Optimal Proposal	0.2406 ± 0.1752	-0.0447 ± 0.0984
Usual PF	0.1149 ± 0.1285	-0.0961 ± 0.0798
Parallel SMC (i)	-0.0434 ± 0.1459	-0.1272 ± 0.0863
Parallel SMC (ii)	0.2760 ± 0.1117	-0.0326 ± 0.0966
Parallel SMC (iii)	0.2584 ± 0.1027	-0.2460 ± 0.1044

Table 4.24: Comparison of Hidden States Estimates for $\sigma_x/\sigma_y = \frac{1}{10}$.

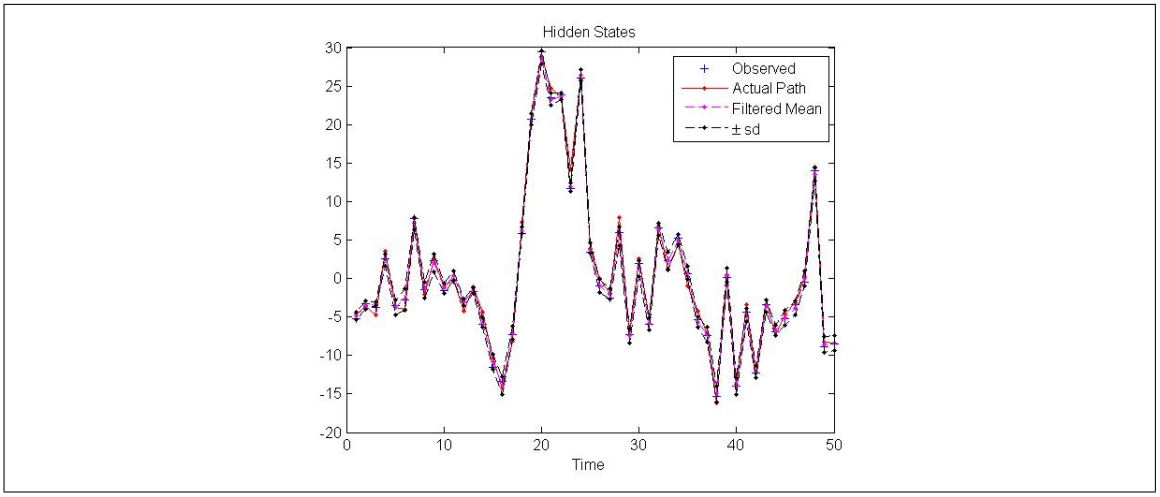

Figure 4.1: Particle trajectory for $\sigma_x/\sigma_y = 10$.

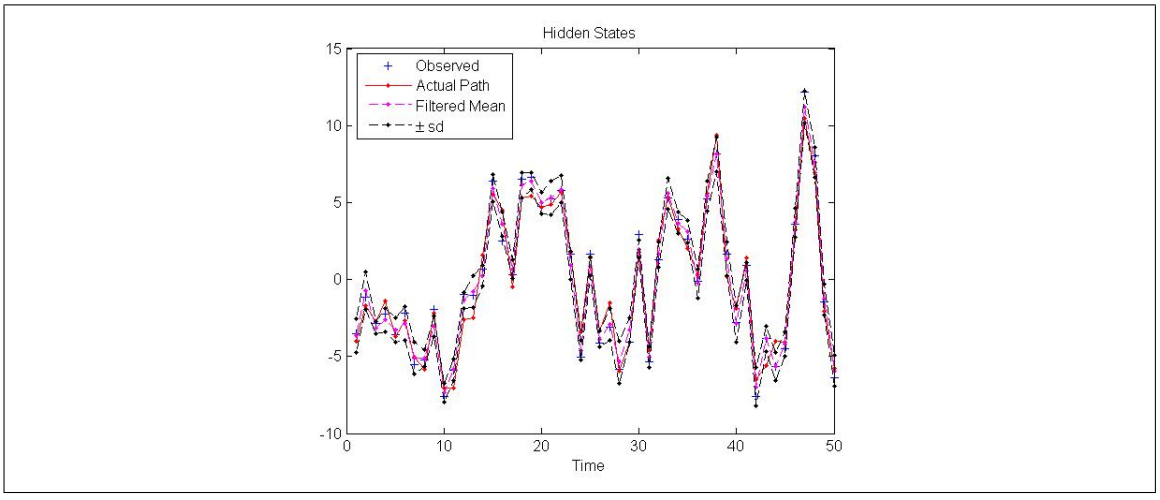

Figure 4.2: Particle trajectory for $\sigma_x/\sigma_y = 5$.

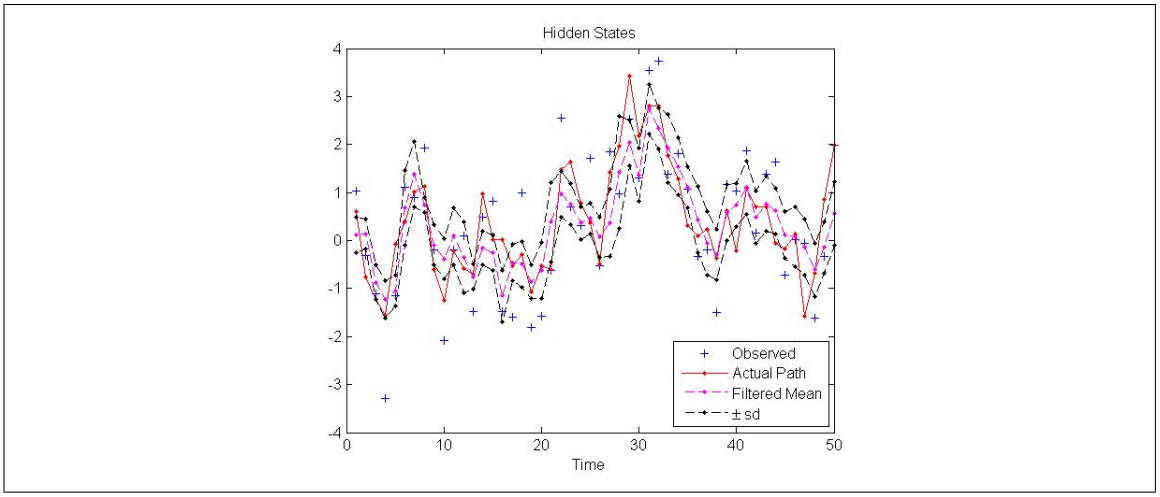

Figure 4.3: Particle trajectory for $\sigma_x/\sigma_y = 1$.

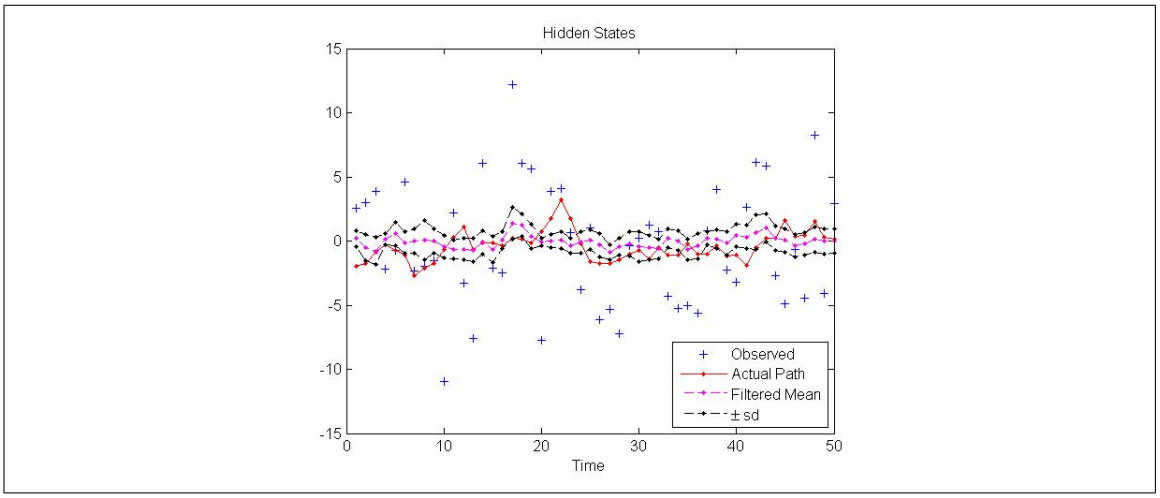

Figure 4.4: Particle trajectory for $\sigma_x/\sigma_y = \frac{1}{5}$.

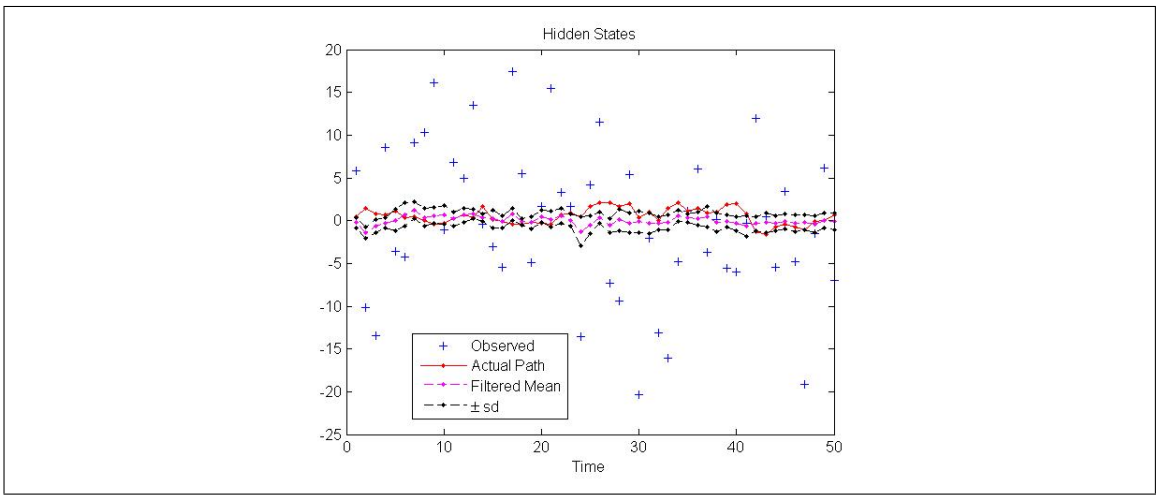

Figure 4.5: Particle trajectory for $\sigma_x/\sigma_y = \frac{1}{10}$.

For high SNR (Figure 4.1 and Figure 4.2), the sample trajectory for the hidden states estimates the actual trajectory well. However, for low SNR (Figure 4.4 and Figure 4.5), the sample trajectory deviates from the actual trajectory significantly. To be specific, from Figure 4.4, the actual value of X_{10} does not lie in the interval based on the mean estimate $\tilde{X}_{10}$. Accordingly, we will expect erratic performance for the usual particle filter, as well as the parallel particle filter in estimating the hidden states for low SNR.

For the estimation of X_{40}, the performance for the usual particle filter and parallel particle filter are comparable. This is expected as the number of distinct values for X_{40} due to resampling for the usual particle filter should be comparable to that from the parallel particle filter.

4.5 Number of subsequences

In our earlier simulation study, we have set $M = 2$ in the simulation for illustration purpose. Recall that the sum (involving the link weights) to be evaluated is

$$\sum_{k_1=1}^{K}\sum_{k_2=1}^{K}\frac{f_\theta(x_{2,1}^{k_2}|x_{1,T}^{k_1})}{f_{\theta 2}(x_{2,1}^{k_2})}.$$

To achieve this naively, one will make use of two 'for' loops to compute the sum, which results in a $\mathcal{O}(K^2)$ computation with a large value for the constant factor. To circumvent this problem, one could make use of vector term by term multiplication. The idea is to create a matrix $\mathbf{A}$ with $a_{ij} = f_\theta(x_{2,1}^j|x_{1,T}^i)/f_{\theta 2}(x_{2,1}^j)$ and sum all the entries. To save computation time, one can create the i-th row, $\mathbf{A}_{i\cdot}$, using vector term by term multiplication of the vectors $\mathbf{a} = (a_1, a_2, \ldots, a_K)'$ and $\mathbf{b} = (b_1, b_2, \ldots, b_K)'$ such that

$$\begin{aligned} a_j &= f_\theta(x_{2,1}^j|x_{1,T}^i), \\ b_j &= 1/f_{\theta 2}(x_{2,1}^j). \end{aligned}$$

That is,

$$\mathbf{A}_{i\cdot} = a_i\mathbf{b}.$$

Therefore, one will only need one for loop to create the matrix **A**. Thereafter, one can simply evaluate the sum of all the entries in the matrix.

The computation of the sum of link weights for the case when $M > 2$ is not significantly more time consuming. If done naively (using 3 'for' loops), the order of the computation will be $\mathcal{O}(K^3)$. However, we could compute this sum using a recursive algorithm. The key idea is to evaluate the sum involving the last subsequence first and proceed in a recursive manner. To elaborate on this, we shall consider the case when $M = 3$. The expression for $\hat{\eta}(\theta)$ is

$$\hat{\eta}(\theta) = \left[\prod_{m=1}^{3}\prod_{t=1}^{T}\bar{w}_{m,t}\right]\left[\frac{1}{K^3}\sum_{k_1=1}^{K}\sum_{k_2=1}^{K}\sum_{k_3=1}^{K}\alpha_{2,T}(X_{1,T}^{k_1}, X_{2,1:T}^{k_2})\alpha_{3,T}(X_{2,T}^{k_2}, X_{3,1:T}^{k_3})\right].$$

For ease of notation, we shall define $\alpha_{m,i,j} = \alpha_{m,T}(X_{m-1,T}^{i}, X_{m,1:T}^{j})$. Thus

$$\hat{\eta}(\theta) = \left[\prod_{m=1}^{3}\prod_{t=1}^{T}\bar{w}_{m,t}\right]\left[\frac{1}{K^3}\sum_{i=1}^{K}\sum_{j=1}^{K}\sum_{k=1}^{K}\alpha_{2,i,j}\alpha_{3,j,k}\right].$$

Concentrating on the sum, we have

$$\sum_{i=1}^{K}\sum_{j=1}^{K}\sum_{k=1}^{K}\alpha_{2,i,j}\alpha_{3,j,k} \;=\; \sum_{i=1}^{K}\sum_{j=1}^{K}\alpha_{2,i,j}\left(\sum_{k=1}^{K}\alpha_{3,j,k}\right).$$

Using matrix computation, we can first compute the vector $\mathbf{c} := (c_1,\ c_2,\ \cdots,\ c_K)'$ such that

$$c_j = \sum_{k=1}^{K}\alpha_{3,j,k}.$$

Using this notation, the sum can be rewritten as

$$\sum_{i=1}^{K}\sum_{j=1}^{K}\alpha_{2,T}(X_{1,T}^{i}, X_{2,1:T}^{j})c_j = \sum_{i=1}^{K}\sum_{j=1}^{K}\frac{f_\theta(x_{2,1}^{j}|x_{1,T}^{i})c_j}{f_{\theta 2}(x_{2,1}^{j})}.$$

One can make use of the same technique in the case of $M = 2$, where we will now create the matrix **B** such that

$$b_{ij} = \frac{f_\theta(x_{2,1}^{j}|x_{1,T}^{i})c_j}{f_{\theta 2}(x_{2,1}^{j})}.$$

One will create the i-th row of **B** using vector term by term multiplication of vectors **a**, **b** and **c** as illustrated earlier. Subsequently, we can sum up the entries of **B** to

evaluate the required sum. Using this recursive technique, one can compute the sum of the link weights with computation time of order $\mathcal{O}(MK^2)$ instead of $\mathcal{O}(K^3)$.

To verify this, we shall run simulation for the case when $T = 60$, $\alpha = 0.9$ and $\sigma_x/\sigma_y \in \{\frac{1}{10}, 1, 10\}$. 10 estimates of the log likelihood will be computed to obtain the mean and standard error of the estimates. For comparison purposes, we shall set $K = K_p = 500$. We will use option (ii) for the parallel particle filter for the various values of M used. We shall denote $\tilde{l}_{2,M}$ to denote the estimate obtained by using parallel particle filter using option (ii) (the exact distribution of X_{mT+1} given $Y_{1:mT}$ for $1 \leq m < M$ is used for $f_{\theta m}$ and M subsequences where $M \in \{1, 2, 3, 4, 5\}$. Note that when $M = 1$, the estimate is obtained using the usual particle filter. l_T denotes the log likelihood function computed using Kalman filtering. We will define,

(i) T_f to be the time taken to complete the particle filter for one subsequence of length T/M;

(ii) T_l to be the time taken to complete the computation of the sum of link weights;

(iii) S to be the mean time taken to complete 1 run.

Table 4.25 to 4.27 summarise the results of this simulation study.

	Estimate	T_f	T_l	S
l_{60}	-227.41			
$\tilde{l}_{2,1}$	-227.37 ± 0.02	–	–	0.3514
$\tilde{l}_{2,2}$	-227.37 ± 0.03	0.1709	0.0493	0.2202
$\tilde{l}_{2,3}$	-227.42 ± 0.03	0.1115	0.0877	0.1992
$\tilde{l}_{2,4}$	-227.46 ± 0.03	0.0834	0.1373	0.2207
$\tilde{l}_{2,5}$	-227.41 ± 0.05	0.0739	0.2117	0.2855

Table 4.25: Comparison of Log Likelihood Estimates and Computational Time for Different Values of M with $\sigma_x/\sigma_y = \frac{1}{10}$.

	Estimate	T_f	T_l	S
l_{60}	-98.81			
$\tilde{l}_{2,1}$	-98.76 ± 0.09	–	–	0.3474
$\tilde{l}_{2,2}$	-98.67 ± 0.10	0.1711	0.0480	0.2191
$\tilde{l}_{2,3}$	-98.95 ± 0.13	0.1081	0.0866	0.1947
$\tilde{l}_{2,4}$	-98.79 ± 0.07	0.0805	0.1349	0.2153
$\tilde{l}_{2,5}$	-98.76 ± 0.08	0.0630	0.1827	0.2457

Table 4.26: Comparison of Log Likelihood Estimates and Computational Time for Different Values of M with $\sigma_x/\sigma_y = 1$.

	Estimate	T_f	T_l	S
l_{60}	-174.66			
$\tilde{l}_{2,1}$	-175.25 ± 0.36	–	–	0.3476
$\tilde{l}_{2,2}$	-175.28 ± 0.37	0.1650	0.0467	0.2116
$\tilde{l}_{2,3}$	-175.49 ± 0.37	0.1091	0.0813	0.1904
$\tilde{l}_{2,4}$	-175.78 ± 0.24	0.0808	0.1303	0.2112
$\tilde{l}_{2,5}$	-175.31 ± 0.32	0.0734	0.1968	0.2702

Table 4.27: Comparison of Log Likelihood Estimates and Computational Time for Different Values of M with $\sigma_x/\sigma_y = 10$.

For the three different values of σ_x/σ_y, the estimates for the different values of M perform reasonably well with reference to the actual log likelihood computed via Kalman filtering. For most estimates, the actual log likelihood will fall within two standard errors of the mean estimate. An interesting finding is that the standard errors for the case when $M = 4$ is smaller compared to other values of M. Since the length of the subsequences decreases with the number of subsequences introduced, one would expect the effect of weight degeneracy to decrease as well. This might suggest that using more subsequences will give a more precise estimate for the log likelihood.

We shall now discuss the computation time involved for different values of M. From earlier remarks and the nature of parallel particle filter, one would expect T_f to be of order $\mathcal{O}(1/M)$ for a fixed value of T while T_l to be of order $\mathcal{O}(M)$ for a fixed value of K. Subsequently, one would expect to have an optimal number of subsequences for the computation time. For our simulation study with $T = 60$ and $K = 500$, the optimal number of subsequences appear to be 3, supported by the values of S. From Table 4.25 to 4.27, we are able to justify that T_f is of order $\mathcal{O}(1/M)$. Using the recursive technique discussed in this section, we are able to achieve T_l to be of order $\mathcal{O}(M)$. We can see that using more subsequences does not give a significant increase in the computational time. On the contrary, if one could find the optimal number of subsequences to use, we can achieve significant computational cost savings. This supports our belief that using the parallel particle filter could achieve computational cost savings, while possibly achieving other advantages as well.

4.6 Remarks on parallel particle filter

As illustrated in the simulations, the choice for the distribution of X_{T+1} (for two subsequences) plays an important role in the performance of the parallel particle filter. We observe that using an estimation for the actual density of X_{T+1} will improve the performance of the parallel particle filter in contrast when an arbitrary choice is made. When an appropriate choice is made, the performance of the estimate (log likelihood or smoothed mean) are comparable to the estimate obtained via the usual

particle filter. Coupled with a good proposal density for the particle filters, one would be able to obtain estimates with good performance while saving computational time. Accordingly, the use of parallel particle filter would be more attractive to the usual particle filter.

We have discussed in Section 4.5 on the recursive technique for the computation of the sum of link weights. Another possible technique for the computation of this sum is to make use of random sampling of the indices involved which is termed as *sub-sampling* as proposed in Briers *et al.* (2005) and Fearnhend *et al.* (2010) . The usefulness of this technique is to reduce the $\mathcal{O}(K^2)$ computational cost in evaluating the estimates. Though the recursive technique that was discussed in Section 4.5 is not computationally expensive, we will have a larger computational complexity due to the evaluation of the iterated sum as M and K increases.

The idea of sub-sampling is to replace the iterated sum with a single sum, thus achieving computational cost savings. We shall illustrate this by using the case when $M = 2$ for the likelihood function estimate. Let $\{(i(v), j(v)) : 1 \leq v \leq V\}$ be selected i.i.d. random variables from β, a positive distribution on $\mathbb{Z}_K^2$. We will now estimate $\eta(\theta)$ by

$$\hat{\eta}^*(\theta) = \left(\prod_{t=1}^{2T} \bar{w}_t\right) (K^2 V)^{-1} \sum_{v=1}^{V} \frac{f_\theta(x_{T+1}^{j(v)} | x_T^{i(v)})}{f_{\theta 2}(x_{T+1}^{j(v)}) \beta(i(v), j(v))}.$$

Since $\hat{\eta}(\theta)$ is unbiased for $\eta(\theta)$, $\hat{\eta}^*(\theta)$ will be unbiased for $\eta(\theta)$ as well. One can choose $V \sim K$ to achieve a computation cost of $\mathcal{O}(K)$. As the sampling of the indices does not depend on the execution of the parallel particle filters, we can improve the performance of $\hat{\eta}^*(\theta)$ with more sampling without having the need to re-run the particle filters.

As an illustration, we run simulations for the case when $\alpha = 0.9$, $\sigma_x/\sigma_y = \frac{1}{10}$ and $M = 2$ using $K = 500$ particles for the usual parallel SMC and $K \in \{500, 550, 600\}$ for the parallel SMC with sub-sampling. We will choose β to be the discrete uniform distribution on $\mathbb{Z}_K^2$ for simplicity and set $V = K$. We shall denote

(i) $\tilde{l}^*_{2,1}$ to be the log likelihood estimator using sub-sampling with $K = 500$,

(ii) $\tilde{l}^*_{2,2}$ to be the log likelihood estimator using sub-sampling with $K = 550$,

(iii) $\tilde{l}^*_{2,3}$ to be the log likelihood estimator using sub-sampling with $K = 600$,

where option (ii) is used for the distribution of X_{T+1}. The results are summarised in Table 4.28.

	Estimate	T_f	T_l	S
l_{60}	-221.75			
$\tilde{l}_{2,2}$	-221.75 ± 0.02	0.1563	0.0434	0.1997
$\tilde{l}^*_{2,1}$	-221.81 ± 0.07	0.1570	0.0070	0.1640
$\tilde{l}^*_{2,2}$	-221.86 ± 0.11	0.1702	0.0076	0.1778
$\tilde{l}^*_{2,3}$	-221.84 ± 0.07	0.1867	0.0091	0.1957

Table 4.28: Comparison of Log Likelihood Estimates and Computational Time using Sub-sampling.

From Table 4.28, using sub-sampling, we could still achieve estimates with performance that are comparable to the computation when all K^2 coalesced particles are used. One would expect the standard error to be higher since less particles (K versus K^2) are used in the computation of the sum of link weights, which is indeed the case from our simulation studies. Although the standard error is higher, the increase is insignificant compared to the actual value of the estimate. For the same number of particles, the time taken for the computation of the sum of link weights for the sub-sampling technique is about 6 times faster than the usual computation. This is even faster when compared with the time taken for the usual particle filter. With this computational time saving, one would be able to make use of more particles for the estimation of the log likelihood function. Referring to Table 4.28 again, even though the time taken for the particle filter will increase with an increase in the number of particles, the total time taken when we use 600 particles for sub-sampling is still less than that for the usual computation with 500 particles. While the standard error in using 600 particles for the sub-sampling scheme is still much higher than that for the

computation without sub-sampling, one would need to weigh the trade-off in computational time saving with this loss of precision for the estimates if the iterated sum is used for the computation. For higher number of subsequences, we would expect more computational cost savings in the computation of the link weights, making sub-sampling an attractive enhancement for the parallel SMC algorithm. Another way to enhance the method is to choose β such that more influential particles will be given higher weights. Therefore, these particles will be chosen more often to obtain more efficient estimates.

Chapter 5

Discrete Time Gaussian SV Model

5.1 Introduction

Particle filters have many applications in real life problems as stated briefly in Chapter 2. In this chapter, we will consider its use and application in financial mathematics: *the discrete time Gaussian stochastic volatility (SV) model*. Pitt and Shephard (1999) are among the first to use the particle filters (PF) in econometrics. They made use of particle filters to provide estimate of the current volatility of a SV model. Since then, much work has been done in enhancing the use of the PF in this model and its generalisations. A review on the use of particle filters in econometrics can be found in Creal (2012).

For an HMM under a Bayesian setting, one would like to obtain an approximation to the posterior distribution of the parameters θ given a set of observation data $y_{1:T}$ using MCMC methods. In Chapter 2, we discussed some MCMC methods that can be applied to an HMM, for example, the particle Markov chain Monte Carlo (PMCMC) method. The use of PMCMC methods in econometrics is gaining attention. Pitt *et al.* (2012) gave a discussion on the use of PMCMC algorithm in econometrics. As mentioned in Section 2.8, one would need to compute unbiased estimates of the likelihood function for the evaluation of the MH acceptance ratio. In this chapter, to illustrate the use of the proposed method in existing methodology, we will imple-

ment the parallel particle filter in obtaining estimates for the likelihood function for a simple discrete time Gaussian SV model. Since the parallel particle filter provides unbiased estimates for the marginal likelihood function, we will be able to implement this algorithm in the PMCMC method. The objective of this chapter is to show that the parallel particle filter will be able to provide reasonable convergence to the PMCMC method used for the real life example with computational cost savings (since the computation of the likelihood estimates is done in parallel). We will also vary the parameters to investigate any possible trends.

We will begin by first providing a quick review of the Gaussian SV model and its variations, and how particle filters are used in this context in Section 5.2. Thereafter, in Section 5.3, we will consider a simplification of the model that was used by Flury and Shephard (2011) for our simulation study using a set of real data that will be elaborated on. In this section, the set up of the parallel particle filter will be discussed as well. Results and plots obtained from the simulation will be provided to support the analysis and inferences done in Section 5.5. We will see that the use of parallel particle filter is useful for the PMCMC algorithm in terms of computational time savings and performance of the PMCMC algorithm.

5.2 Stochastic volatility model

Stochastic volatility (SV) models are widely used for the modelling and prediction of time-varying volatility on financial markets. These models are important in risk management, asset pricing and asset allocation. Even though stochastic volatility is modelled in a continuous-time setting in practice, the fact that observations are typically recorded at discrete points in time means that the discrete counterpart of SV models are essential. In this section, we shall introduce the standard SV model and some of its extension. For our numerical study, we will base mainly on the standard SV model for simplicity.

5.2.1 The standard stochastic volatility model

Let y_t denotes the log return at time t and h_t denotes the log volatility at time t, where $1 \leq t \leq T$. The standard SV model is given by

$$y_t = \exp(h_t/2)u_t, \tag{5.2.1}$$

$$h_t = \mu + \phi(h_{t-1} - \mu) + \eta_t, \tag{5.2.2}$$

where $u_t \sim N(0, 1)$ and $\eta_t \sim N(0, \sigma_\eta^2)$ are white noise sequences and ϕ in the persistence parameter such that $|\phi| < 1$. If $h_0 \sim N(\mu, \sigma_\eta^2/(1-\phi^2))$, the stationary distribution of h_t is given by $N(\mu, \sigma_\eta^2/(1-\phi^2))$.

Under a Bayesian framework with parameter $\theta = (\mu, \phi, \sigma_\eta^2)$, the likelihood function $p(y_{1:T}|\theta)$ cannot be evaluated in closed form. Thus, numerical methods are necessary to estimate the likelihood function using particle filters or MCMC techniques.

5.2.2 The SV$_t$ model

The standard SV model will not be able to explain the heavy-tail property commonly present in financial return distributions. Accordingly, one will need to make extension to the standard SV model to model this.

A simple generalisation of the standard SV model is to consider a t-distribution for the log return white noise sequence. Specifically, we have the same model in the previous subsection given by (5.2.1) and (5.2.2) with u_t following a t-distribution with v degrees of freedom and $\eta_t \sim N(0, \sigma_\eta^2)$. This is termed as the SV$_t$ model, stressing the t-distribution used in this model. This model can be represented using scale mixture of normal distributions. Let λ_t denote an i.i.d. random variable following an inverse-Gamma distribution. The SV$_t$ model can be expressed as

$$y_t = \exp(h_t/2)\sqrt{\lambda_t}u_t,$$

$$h_t = \mu + \phi(h_{t-1} - \mu) + \eta_t,$$

where $\lambda_t \sim IG(v/2, v/2)$ and $u_t \sim N(0, 1)$ and $\eta_t \sim N(0, \sigma_\eta^2)$ are white noise sequences. λ_t is a hidden variable in this model.

The behaviour of the SV process will be affected due to a change in the distribution of the return process. For a standard SV model, large values of $|y_t|$ induce large values of h_t. However, for a SV_t model, due to the presence of λ_t, large values of $|y_t|$ do not necessary need to increase the value of h_t.

5.2.3 The SV model with jump components

Another shortcoming of the standard SV model is its inability to model possible jumps in the return process. One will be able to include a jump component in the return process (5.2.1) for the standard SV model. The resulting model is called the SVJ model. For this model,

$$\begin{aligned} y_t &= k_t q_t + \exp(h_t/2)u_t, \\ h_t &= \mu + \phi(h_{t-1} - \mu) + \eta_t, \end{aligned}$$

where q_t are i.i.d. Bernoulli random variable taking the value 1 with probability κ, k_t is the jump size that follows $N(\alpha_k, \beta_k)$. For this model, h_t, k_t and q_t are hidden variables.

For the SVJ model, large values of $|y_t|$ will be accounted by the jump process rather than the volatility process.

5.2.4 SV model with leverage

Another generalisation of the standard SV model is to remove the assumption that the return process and volatility process white noise sequences are independent. This can be achieved by introducing a leverage term in the model, which was proposed by Yu (2005). In financial term, leverage is any technique to multiply gains and losses. A basic SV model with leverage is given by

$$\begin{aligned} y_t &= \exp(h_t/2)u_t, \\ h_t &= \mu + \phi(h_{t-1} - \mu) + \eta_t, \end{aligned}$$

where

$$\begin{pmatrix} u_t \\ \eta_t \end{pmatrix} \sim N(\mathbf{0},\, \Sigma), \text{ where } \Sigma = \begin{pmatrix} 1 & \rho \\ \rho & \sigma_\eta^2 \end{pmatrix},$$

and ρ is the correlation between u_t and η_t. When $\rho < 0$, a drop in the return followed by an increase in volatility is known as the leverage effect. The leverage effect can be explained by using financial leverage. When stock prices fall, financial leverage will increase and thus leading to an increase in stock return volatility. This explains the negative correlation between the stock volatility and the stock returns.

5.3 The chosen model

For our simulation study, we will consider a variant of the standard SV model introduced in the previous section. The model is introduced by Flury and Shephard (2011). However, we will make some simplification to it. For this model, the stock log returns, y_t, are assumed to follow

$$y_t = \mu + \nu_t \exp(\beta_0 + \beta_1 \alpha_t)$$

and the stochastic volatility factor

$$\alpha_{t+1} = \phi \alpha_t + \xi_t,$$

where $\xi_t \sim N(0,1)$ and $\nu_t \sim N(0,1)$ are independent white noises. We will set $\alpha_0 \sim N\left(0, 1/(1-\phi^2)\right)$. Thus the stationary distribution of the stochastic volatility factor is $N\left(0, 1/(1-\phi^2)\right)$. In the stated model by Flury and Shephard (2011), a SV model with leverage is considered, that is,

$$\begin{pmatrix} \nu_t \\ \xi_t \end{pmatrix} \sim N(\mathbf{0},\, \Sigma), \text{ where } \Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}.$$

In our study, however, we shall set $\rho = 0$ for simplicity. One can view this model as a special case of a combination of the SVJ model (where μ is the constant jump at each t) and the SV_t model (where $\exp(\beta_0)$ is the variable). For this model, the parameters involved are $\theta = (\mu,\, \beta_0,\, \beta_1,\, \phi)'$. We will use the particle Markov Chain Monte Carlo method (PMCMC) to obtain estimates of the posterior distribution of θ.

5.3.1 Setup of the parallel particle filter

In this model, the exact likelihood is not available. To illustrate our proposed method, we will use the parallel particle filter to compute unbiased estimates of the likelihood to be used for the computation of the acceptance ratio in the PMCMC algorithm.

To simplify the process, we will consider segmenting the observed sequence $\{y_t\}_{t=1}^{2T}$ into two subsequences of equal length. Particle filters with K particles will be run in parallel for each subsequence. For the first subsequence, the distribution $N(0, 1/(1-\phi^2))$ is chosen as the initial distribution of α_1.

Since the exact distribution of $\alpha_{T+1}|y_{1:T}$ is not readily available, one will need to estimate this distribution for the implementation of the particle filter for the second subsequence as discussed in Chapter 4. With reference to details found in Chapter 4, we will make use of particle filtering with K particles on the observation sequence $\{y_{T-r}, \ldots, y_T\}$ to obtain an estimate of this distribution with $r = 10$.

5.3.2 Parameter proposal

We shall assume a Gaussian prior given by $\theta \sim N(\theta_0, I_4)$ where

$$\theta_0 = (0.036,\ -0.286,\ 0.077,\ 0.984)'.$$

As stated by Flury and Shephard (2011), any proposal for $\phi \notin (-1, 1)$ will be rejected. We will be using the following random walk proposals for $\theta_i = (\mu_i, \beta_{0,i}, \beta_{1,i}, \phi_i)$

$$\begin{aligned}
\mu_i &= \mu_{i-1} + a_1\eta_{1,i}, \\
\beta_{0,i} &= \beta_{0,i-1} + a_2\eta_{2,i}, \\
\beta_{1,i} &= \beta_{1,i-1} + a_3\eta_{3,i}, \\
\phi_i &= \phi_{i-1} + a_4\eta_{4,i},
\end{aligned}$$

where $\eta_{j,i} \sim N(0,1)$ are independent for $j = 1, \ldots, 4$ and $i = 1, \ldots, N$ and a_j are the tuning parameters. The tuning parameters are chosen to have an acceptance probability of around 0.4 to 0.5. Gibbs sampling will be done at each iteration to update

the parameters, hidden states and likelihood function.

5.3.3 Chosen data and setup

The data we have chosen for this simulation is the end-of-day adjusted level of the S&P 500 (GSPC) index. The data are obtained from the online database Quandl (`https://www.quandl.com/YAHOO/INDEX_GSPC-S-amp-P-500-Index`). We will vary the number of daily observations from the set of data from 03.01.1995 until 17.10.1995, excluding the non-trading days. At 03.01.1995, we will set $t = 0$. The daily returns are defined as

$$y_t = 100(\log \text{SNP500}_t - \log \text{SNP500}_{t-1}),$$

where SNP500_t denotes the end-of-day adjusted level of the index at time t. For the set of data, we will have 200 observations for y_t. We shall consider the cases when $2T \in \{50, 100, 200\}$. We will vary the number of particles $K \in \{100, 300, 500\}$ used for the parallel particle filters to examine the effect on the PMCMC algorithm. We will run the simulation to obtain chains of length $N = 1000$.

We will report the means and standard errors for the four prior parameters. The posterior covariance, correlation and acceptance probability will be reported as well. Sample paths of the four parameters will be given, as well as the autocorrelation plots. Running average plots of the parameters for different setup will be included to assess the burn-in periods. Plots of the log likelihood estimates against θ_i are included to assess the fluctuations.

5.4 Tables of simulations

We shall first report the results of the numerical study done. The analysis of these result will be discussed in Section 5.5. The results are calculated based on a burn-in period of 200. The choice of this value for the burn-in period will be discussed in Section 5.5 as well. For Tables 5.1 to 5.9, the posterior correlations are above the

leading diagonal, while the posterior covariances are below the leading diagonal.

	Mean	Acceptance Rate	Posterior Covariance and Correlation			
μ	-0.0089	0.471	0.0081	-0.0157	0.0302	-0.1565
β_0	-0.5250	0.401	-0.0002	0.0136	-0.1015	0.1294
β_1	0.0707	0.464	0.0007	-0.0030	0.0624	-0.0228
ϕ	0.0291	0.458	-0.0068	0.0073	-0.0027	0.2316

Table 5.1: Result for PMCMC algorithm using $2T = 50$, $K = 100$.

	Mean	Acceptance Rate	Posterior Covariance and Correlation			
μ	-0.0057	0.498	0.0096	-0.0774	0.0019	-0.1772
β_0	-0.5237	0.424	-0.0010	0.0164	-0.1015	0.1225
β_1	0.0286	0.458	0.0000	-0.0028	0.0451	0.0417
ϕ	0.1150	0.429	-0.0087	0.0079	0.0045	0.2538

Table 5.2: Result for PMCMC algorithm using $2T = 50$, $K = 300$.

	Mean	Acceptance Rate	Posterior Covariance and Correlation			
μ	0.0102	0.494	0.0067	0.0160	0.0224	-0.1351
β_0	-0.5155	0.414	0.0002	0.0235	-0.0964	0.2723
β_1	0.0504	0.483	0.0004	-0.0034	0.0519	-0.0132
ϕ	0.0780	0.465	-0.0058	0.0220	-0.0016	0.2767

Table 5.3: Result for PMCMC algorithm using $2T = 50$, $K = 500$.

	$K = 100$		$K = 300$		$K = 500$	
Parameter	Mean	NSE	Mean	NSE	Mean	NSE
μ	0.0089	0.0032	-0.0057	0.0034	0.0102	0.0029
β_0	-0.5250	0.0041	-0.5237	0.0045	-0.5154	0.0054
β_1	0.0707	0.0088	0.0286	0.0075	0.0504	0.0081
ϕ	0.0291	0.0170	0.1150	0.0178	0.0780	0.0186

Table 5.4: Geweke estimate for the mean and numerical standard error of the parameters for $2T = 50$.

Parameter	Lag 1	Lag 5	Lag 10	Lag 50
μ	0.655	0.201	0.014	0.045
β_0	0.721	0.196	0.006	−0.018
β_1	0.747	0.239	0.148	0.027
ϕ	0.739	0.244	0.096	0.013

Table 5.5: Autocorrelation for $2T = 50$, $K = 500$ and $N = 1000$.

	Mean	Acceptance Rate	Posterior Covariance and Correlation			
μ	0.0151	0.424	0.0030	−0.1133	0.1950	−0.0753
β_0	−0.6573	0.438	−0.0006	0.0106	−0.2469	0.2633
β_1	0.0984	0.454	0.0027	−0.0065	0.0662	−0.2092
ϕ	0.1792	0.446	−0.0019	0.0127	−0.0252	0.2195

Table 5.6: Result for PMCMC algorithm using $2T = 100$, $K = 100$.

	Mean	Acceptance Rate	Posterior Covariance and Correlation			
μ	0.0265	0.440	0.0028	−0.0894	0.0129	−0.1335
β_0	−0.6661	0.438	−0.0005	0.0096	0.0699	0.2186
β_1	0.0485	0.454	0.0002	0.0019	0.0760	0.0173
ϕ	0.0895	0.444	−0.0033	0.0101	0.0023	0.2251

Table 5.7: Result for PMCMC algorithm using $2T = 100$, $K = 300$.

	Mean	Acceptance Rate	Posterior Covariance and Correlation			
μ	0.0235	0.435	0.0023	−0.0874	0.0665	−0.1705
β_0	−0.6704	0.543	−0.0005	0.0126	−0.3366	0.1787
β_1	0.1239	0.488	0.0008	−0.0099	0.0682	−0.1625
ϕ	0.1475	0.456	−0.0042	0.0102	−0.0215	0.2573

Table 5.8: Result for PMCMC algorithm using $2T = 100$, $K = 500$.

	$K = 100$		$K = 300$		$K = 500$	
Parameter	Mean	NSE	Mean	NSE	Mean	NSE
μ	0.0151	0.0019	0.0265	0.0019	0.0235	0.0017
β_0	-0.6573	0.0036	-0.6661	0.0034	-0.6704	0.0040
β_1	0.0984	0.0091	0.0485	0.0097	0.1239	0.0092
ϕ	0.1792	0.0166	0.0895	0.0168	0.1475	0.0179

Table 5.9: Geweke estimate for the mean and numerical standard error of the parameters for $2T = 100$.

Parameter	Lag 1	Lag 5	Lag 10	Lag 50
μ	0.582	0.113	0.022	-0.043
β_0	0.776	0.304	0.121	0.036
β_1	0.755	0.337	0.166	0.003
ϕ	0.773	0.324	0.160	-0.094

Table 5.10: Autocorrelation for $2T = 100$, $K = 500$ and $N = 1000$.

	Mean	Acceptance Rate	Posterior Covariance and Correlation			
μ	0.0444	0.408	0.0006	0.0829	0.2350	-0.2566
β_0	-0.9046	0.395	0.0004	0.0324	0.1840	-0.1491
β_1	0.1213	0.347	0.0002	0.0013	0.0015	-0.6703
ϕ	0.9462	0.347	-0.0002	-0.0010	-0.0009	0.0013

Table 5.11: Result for PMCMC algorithm using $2T = 200$, $K = 100$.

	Mean	Acceptance Rate	Posterior Covariance and Correlation			
μ	0.0459	0.443	0.0007	-0.1595	0.1935	-0.1277
β_0	-0.8662	0.498	-0.0006	0.0198	-0.2236	0.2763
β_1	0.1261	0.484	0.0007	-0.0046	0.0214	-0.6592
ϕ	0.8855	0.407	-0.0004	0.0041	-0.0102	0.0111

Table 5.12: Result for PMCMC algorithm using $2T = 200$, $K = 300$.

	Mean	Acceptance Rate	Posterior Covariance and Correlation			
μ	0.0471	0.428	0.0004	−0.0109	−0.1471	0.1512
β_0	−0.7143	0.441	0.0000	0.0549	0.0080	0.4070
β_1	0.1022	0.447	−0.0001	0.0000	0.0004	−0.2211
ϕ	0.9684	0.550	0.0001	0.0020	−0.0001	0.0004

Table 5.13: Result for PMCMC algorithm using $2T = 200$, $K = 500$.

	$K = 100$		$K = 300$		$K = 500$	
Parameter	Mean	NSE	Mean	NSE	Mean	NSE
μ	0.0444	0.0008	0.0459	0.0009	0.0471	0.0007
β_0	−0.9046	0.0064	−0.8661	0.0049	−0.7143	0.0083
β_1	0.1213	0.0014	0.1261	0.0052	0.1022	0.0007
ϕ	0.9462	0.0013	0.8855	0.0037	0.9684	0.0007

Table 5.14: Geweke estimate for the mean and numerical standard error of the parameters for $2T = 200$.

Parameter	Lag 1	Lag 5	Lag 10	Lag 50
μ	0.941	0.718	0.538	0.058
β_0	0.979	0.905	0.823	0.275
β_1	0.929	0.668	0.467	−0.146
ϕ	0.982	0.923	0.845	0.347

Table 5.15: Autocorrelation for $2T = 200$, $K = 500$ and $N = 1000$.

$2T$	Burn	Total	Min. Length	Dependence Factor
50	9	2336	937	2.493
100	12	2959	937	3.158
200	21	6561	937	7.002

Table 5.16: Raftery-Lewis diagnostic for different values of $2T$ with $K = 500$, $N = 1000$.

Parameter	$2T = 50$	$2T = 100$	$2T = 200$
μ	0.871	0.463	0.000
β_0	0.502	0.024	0.000
β_1	0.269	0.399	0.000
ϕ	0.190	0.340	0.000

Table 5.17: Geweke Chi-Square significance for $K = 500$ and burn-in period = 500.

5.5 Analysis of simulation results

In this section, we will analyse the results obtained from the numerical study of the chosen SV model. The plots obtained from the simulation are provided in Section 5.7. We will first give a discussion on the burn-in period for the different setup of the simulation parameters. Thereafter, the performance of the PMCMC algorithm using the parallel particle filter will be assessed for different values of $2T$.

5.5.1 Burn-in period

One way to assess the chain for proper mixing is the use of a running mean plot. The running mean plots for the case when $K = 300$ particles are used are given in Figures 5.1 to 5.3. For the case when $2T = 50$ (Figure 5.1, the mean for most of the parameters stabilise after 200 iterations. The same comment can be said for the case when $2T = 100$ (Figure 5.2). From these plots, we will then set the burn-in period to be 200 for these cases.

When $2T = 200$ (Figure 5.3), the running mean plots of most of the parameters stabilise after 200 iteration as well. One exception is the running mean plot of β_1. After 800 iterations, there is a significant dip in the running mean value which might suggest that the chain has not attained its stationary distribution. However, although the chain might not have mixed properly, we will set the burn-in period to be 200 for consistency with the other cases.

5.5.2 Performance of algorithm for $2T = 50$

The results for the case when $2T = 50$ are summarised in Tables 5.1 to 5.5. We shall first discuss the inferences by observing the plots obtained from the simulation.

From Figure 5.1, we observe that for the case when $2T = 50$, using a burn-in time of 200 works for most of the parameters (except for ϕ). The sample plots for the parameters when $K = 100$ (Figure 5.4), $K = 300$ (Figure 5.6) and $K = 500$ (Figure 5.8) indicate proper mixing of the chains as well.

The ACF plots (Figure 5.5, Figure 5.7 and Figure 5.9) for the four parameters for different value of K reflect no significant spikes in the plots apart for lag 1. The peaks of the ACF plots for higher lag values are contained in the interval $(-0.2, 0.2)$, indicating a low level of correlation within the chain. This further suggests that the chains are well mixed for the different values of K chosen. Thus, even though the number of particles used are low, we are able to achieve a fast convergence to the target posterior distributions.

Referring to Table 5.5, the autocorrelation at the various lag checkpoints are low for $K = 500$, affirming that the chains have low degree of correlation within the chains. Thus the chains are reasonably well-mixed.

From Table 5.4, the estimates of the mean of the parameters indicate fluctuations for different values of K. This might suggest that for smaller number of particles used, the chain has to be run for a longer duration to ensure that the chains give good approximation to the posterior distributions.

Table 5.4 reports the Geweke estimate for the numerical standard error (NSE) for different values of K. From the table, the NSE of the parameters show slight fluctuations as K increases from 100 to 500. This suggests that the number of particles does not affect the NSE for the case when $2T = 50$.

From Tables 5.1 to 5.3, the estimates of the posterior covariance of the parameters show slight fluctuation for different values of K. This might suggest that when the length of the observation sequence is short, the number of particles needed for the parallel particle filter can be smaller to achieve a reasonable convergence to the target posterior distributions.

For different values of K, we have relatively higher posterior correlation coefficient for μ and ϕ; β_0 and ϕ. However, the absolute values of these coefficients are low (less than 0.3), suggesting that the parameters of the SV model are uncorrelated when $2T = 50$.

Consequently, for this setup, the PMCMC algorithm using the parallel particle filter perform reasonably well in approximating the posterior distributions of the parameters of the HMM.

5.5.3 Performance of algorithm for $2T = 100$

The results for the case when $2T = 100$ are summarised in Tables 5.6 to 5.10. We will first consider the findings from the plots obtained from the simulation.

From Figure 5.2, we observe that for the case when $2T = 100$, choosing the burn-in time to be 200 for most of the parameters (except for ϕ) is reasonable. The sample paths for the different values of K are given in Figure 5.10, Figure 5.12 and Figure 5.14 while the ACF plots are given in Figure 5.11, Figure 5.13 and Figure 5.15. Referring to the sample paths plots, one would be able to see that the PMCMC algorithm using the parallel particle filter is able to provide fast convergence to the target posterior density, where fast mixing of the chain is evident. Thus the performance of the PMCMC algorithm is satisfactory given the fast convergence to the target densities.

The ACF plots do not give significant hikes apart from lag 1. The peaks of the plots

are contained in the interval $(-0.2, 0.2)$, suggesting that the chains produced are uncorrelated. There are no significant patterns to suggest that there is a periodic trend within the chains produced. Based on the ACF plots, we can deduce that the chains exhibit fast mixing for the different value of K used in the simulation.

Referring to Table 5.10, the autocorrelation at the various lag checkpoints are low for $K = 500$, affirming that the chains have low degree of correlation within the chains. Thus the chains are reasonably well-mixed for this case.

From Table 5.9, the estimates of the mean of the parameters fluctuates as K increases from 100 to 500. Thus the estimates of the mean are sensitive to the number of particles used in the parallel particle filter for a fixed iteration. As in the case where $2T = 50$, this might suggest that the chains have to be run for a larger number of iterations to ensure that the chains give good approximation to the posterior distributions if the number of particles used is small.

Table 5.9 reports the Geweke estimate for the numerical standard error (NSE) for different values of K. From the table, the NSE of the parameters show small fluctuations as K increases from 100 to 500. This suggests that the number of particles does not affect the NSE for the case when $2T = 100$ as well.

From Tables 5.6 to 5.8, the estimates of the posterior covariance of the parameters show slight fluctuation for different values of K. This might suggest that for this length of the observation sequence, one need not use a large number of particles for the parallel particle filter to achieve reasonable convergence to the target posterior distributions.

For different values of K, we have relatively higher posterior correlation coefficient for β_0 and β_1; β_0 and ϕ. This is not surprising due to the model used, where correlation between β_0 and β_1, β_0 and ϕ are to be expected. However, the absolute values of these coefficients are low (less than 0.3 for most cases), suggesting that the parameters of

the SV model are uncorrelated when $2T = 100$.

Thus, for the case when $2T = 100$, the PMCMC algorithm using the parallel particle filter perform reasonably well in approximating the posterior distributions of the parameters of the HMM as well.

5.5.4 Performance of algorithm for $2T = 200$

The results for the case when $2T = 100$ are summarised in Tables 5.11 to 5.15. As with the other case, we shall first consider the findings from the plots obtained from the simulation.

From Figure 5.3, we observe that for the case when $2T = 200$, the plots justify the choice of setting the burn-in time to be 200 for most of the parameters (except for ϕ). The sample paths for the different values of K are given in Figure 5.16, Figure 5.18 and Figure 5.20 while the ACF plots are given in Figure 5.17, Figure 5.19 and Figure 5.21. Referring to the sample paths plots, when 100 particles are used (Figure 5.16), the sample paths showed slow mixing of the chain. The plots for 300 particles (Figure 5.18) are more encouraging where mixing of the chains are evident. When 500 particles are used (Figure 5.18), the parameters μ and β_1 show reasonable mixing while the parameters β_0 and ϕ show slow mixing. This suggests that for a larger set of observed data, one will need more particles for the parallel particle filter in order to get reasonable estimates for the log likelihood function. For this setup, one might need to use more particles to achieve better mixing.

The ACF plots reiterate the points mentioned earlier. For the case when 100 particles are used (Figure 5.17), the peaks of the ACF plots exceeded the ideal range of $(-0.2, 0.2)$. The high peaks indicated that the chains are poorly mixed and exhibited high correlation when a small number of particles are used for the parallel particle filter. However, when 300 particles are used (Figure 5.19), the results obtained from the ACF plots are better. The peaks are now contained in the interval $(-0.2, 0.2)$,

which support the fact that the chains are well mixed and uncorrelated. For the case when 500 particles are used (Figure 5.21), the ACF plots of β_0 and β_1 indicates high peaks at large lag values. This suggests that the chains exhibit a high degree of correlation, indicating that the chains are not well-mixed. Consequently, one will need to run the chain for a larger number of iterations in order to ensure that the chains have mixed properly.

Referring to Table 5.15, the autocorrelation at the various lag checkpoints are relatively high for $K = 500$. This illustrates that the chains have a high degree of correlation within the chains at lag 50. Thus the chains are not well-mixed and a larger iteration is required to achieve well-mixing.

From Table 5.14, the estimates of the mean of the parameters fluctuates as K increases from 100 to 500. As with the previous cases, the estimates of the mean are sensitive to the number of particles used in the parallel particle filter for a fixed iteration. This might suggest that the chains have to be run for a larger number of iterations to ensure that the chains give good approximation to the posterior distributions for small number of particles used.

Table 5.14 reports the Geweke estimate for the numerical standard error (NSE) for different values of K. From the table, the NSE of the parameter μ and β_0 show small fluctuations as K increases from 100 to 500. However, the NSE of the parameter β_1 and ϕ for the case when $K = 300$ differs significantly from the other two values of K used. Since the NSE of the parameters are close for most cases, we may still conclude that the number of particles does not affect the NSE for the case when $2T = 200$.

From Tables 5.11 to 5.13, the estimates of the posterior covariance of the parameters differ significantly for different values of K. This might suggest that for this length of the observation sequence, a larger number of particles for the parallel particle filter should be used to achieve a reasonable convergence to the target posterior distributions. Another possible solution is to run the chains for a larger number of iterations

to ensure proper mixing.

For different values of K, we have relatively higher posterior correlation coefficient for β_0 and ϕ; β_1 and ϕ. The high value for the correlation coefficients might be due to poor mixing of the chains produced. Thus, either a larger number of particles need to be used for the parallel particle filter or a larger number of iterations need to be set to achieve a better result.

Consequently, for the case when $2T = 200$, the performance of the PMCMC algorithm using the parallel particle filter is not satisfactory. This could be due to the small number of iterations used for the simulation. In order to achieve a better performance, one will need to use a larger number of iterations to ensure proper mixing of the chains. Better approximation to the posterior distributions can thus be achieved.

5.5.5 Effect of T on the chain-mixing

We will make use of two diagnostics to assess the convergence of the chains obtained from the PMCMC algorithm to the stationary distributions for different values of $2T$. The diagnostics that we will be using are the *Raftery-Lewis diagnostic* and the *Geweke diagnostic*. We will discuss the diagnostic briefly before the analysis of our result.

Raftery-Lewis diagnostic

Raftery and Lewis (1992) introduced a run length control diagnostic based on a criterion of accuracy of estimation of the quantile q of the stationary distribution. It is intended for use on a short pilot run of a Markov chain. The diagnostic consists of the following: number of burn-in, required number of iterations, minimum length and the dependence factor. We shall elaborate on these quantities.

The number of 'burn-in' refers to the number of iterations to be discarded at the beginning of the chain. The number of iterations required to estimate the quantile q

to within an accuracy of $\pm r$ with probability p is calculated. The minimum length is the required sample size for a chain with no correlation between consecutive samples. The dependence factor, I, for the Raftery-Lewis diagnostic measures the proportional increase in the number of iterations attributable to serial dependence. A dependence factor larger than 5 is a cause of concern. This might suggest a high correlation between the parameters or poor mixing of the chains.

Table 5.16 summarises the Raftery-Lewis diagnostic for $2T \in \{50, 100, 200\}$ using $K = 500$ particles for the parallel particle filter and running the chain for $N = 1000$ iterations. The diagnostic are computed based on $q = 0.025$, $r = 0.01$ and $p = 0.95$. The number of burn-in necessary for the algorithm increases as T increases, indicating that the chains have to be run longer in order to achieve proper mixing. The number of necessary iterations for the chains is given by *Total* in the table. As T increases, the number of necessary iterations for the chains increases. This is in accordance with the findings from the earlier subsections, where a larger number of iterations is required to ensure good mixing for the chains as T increases.

From the table, the dependence factor when $2T = 50$ and $2T = 100$ are less than 5. This suggests that the chains for these cases achieved proper mixing. The dependence factor when $2T = 200$ is higher than 5. This might indicate that the chains are not well-mixed, warranting a larger number of iterations for this case.

Geweke diagnostic

Geweke (1992) proposed a convergence diagnostic for Markov chains. The procedure considers the first and last part of a Markov chain to determine whether they can be considered as coming from the same distribution. This test statistic is based on the sample means and the asymptotic variance (estimated using spectral density methods) of the first and last part of a Markov chain.

Table 5.17 summarises the Geweke chi-square significance for $2T \in \{50, 100, 200\}$ using $K = 500$ particles for the parallel particle filter. A burn-in period of 500 is

chosen for the computations of the significance. The Geweke diagnostic is computed by comparing the means of the first 20 percent and the last 50 percent of the Markov chain. A high value for the significance suggests that the chain has reached its stationary distribution.

From the table, with a level of significance of 0.05, we can conclude that the relevant means for the case when $2T = 50$ are the same for all the parameters. This indicates that the chains have reached its stationary distributions. For the case when $2T = 100$, except for the parameter β_0, we can conclude that the means are equal with a level of significance of 0.05 as well. However, for the case when $2T = 200$, the Geweke chi-square significance indicates that the means are different for all the parameters. This reinforces the fact that a larger number of iterations is needed for this case to ensure proper mixing and convergence to the target posterior distributions.

5.5.6 Remarks on log likelihood plots

The plots for the log likelihood functions for the different settings are provided in Figure 5.24 to Figure 5.32 to illustrate the distribution of the log likelihood as the chain propagates. When $2T = 50$ (Figures 5.24 to 5.26), the log likelihood value oscillates within an interval of length 10 with a short burn-in period. We have similar results for the burn-in period and the interval width for the case when $2T = 100$ (Figures 5.27 to 5.29) and when $2T = 200$ (Figures 5.30 to 5.32). This suggests that the parallel particle filter provides a reasonable estimate for the log likelihood as the Markov chains propagates with time.

To summarise our findings, from the simulation studies, we can see that the parallel particle filter is useful for the estimation of likelihood function in the PMCMC algorithm. Apart from computational cost savings, the convergence of the Markov chains are satisfactory with a short burn-in period for the case when $2T = 50$ and $2T = 100$ using a relatively small number of particles ($K < 500$). Recall that only 2 subsequences are used in the computation and one could achieve additional computational

cost savings if more subsequences are used.

While the case for $2T = 200$ is unsatisfactory, it could be due to the small number of iterations used for the simulation study. Using more particles and increasing the number of iterations will possibly improve the performance of the PMCMC algorithm. To illustrate this point, we shall examine the running mean plot with $N = 10000$ for the case when $2T = 200$, where $K = 300$ particles is used for the parallel particle filter, given in Figure 5.22. The plot suggests a burn-in period of 2000 for this case since the running mean for the parameters stabilises after this number of iteration. The ACF plots (Figure 5.23) exhibits a low degree of correlation within the chains as the hikes are contained in the interval $(-0.2, 0.2)$. We could see that running the chains longer will give a better approximation to the posterior distributions.

5.6 Remarks

In this chapter, we have illustrated the implementation of parallel particle filter to obtain unbiased estimates of the likelihood function that will be used in the PMCMC algorithm. We have considered a SV model with real data to show the use of this algorithm. We shall remarked that the performance of the PMCMC algorithm will be affected by the model chosen. As mentioned, one would need to consider the optimal number of particles to be used in the parallel particle filter and an appropriate length of iterations to gain satisfactory performance for the PMCMC algorithm. On another note, one might be able to get better performance using the algorithm with an extension of the proposed model. For instance, we might include the leverage factor in the model which was omitted due to simplicity.

5.7 Plots

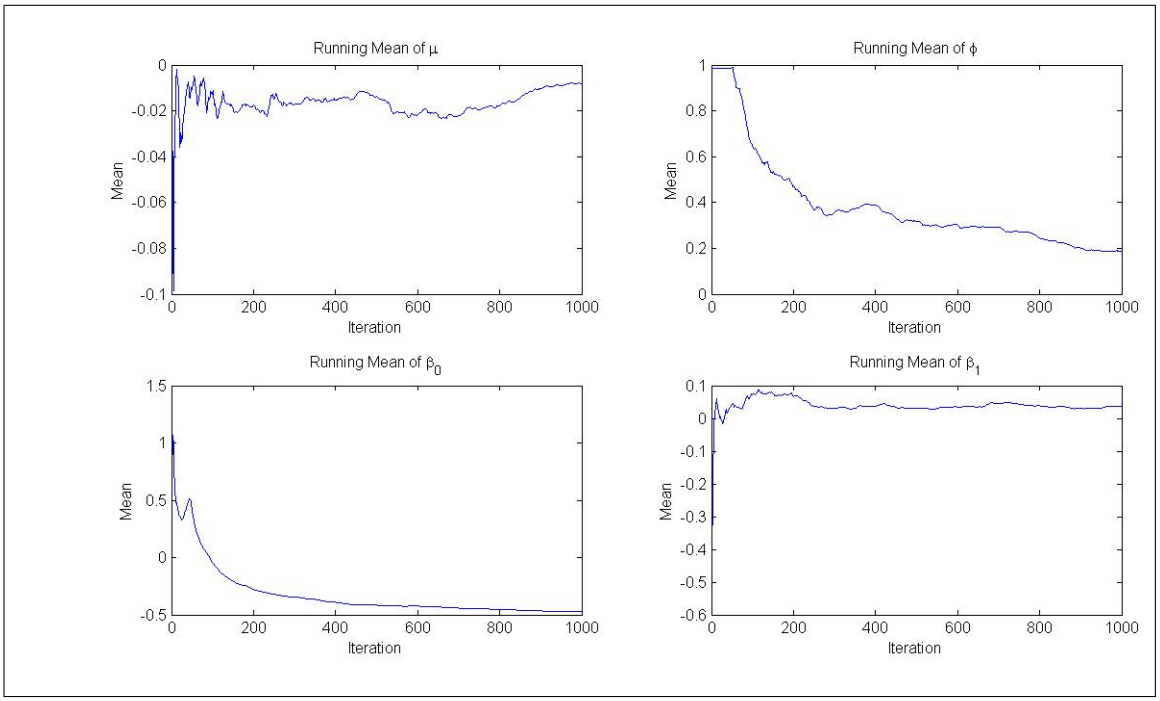

Figure 5.1: Running Mean of Parameters for $2T = 50$, $K = 300$.

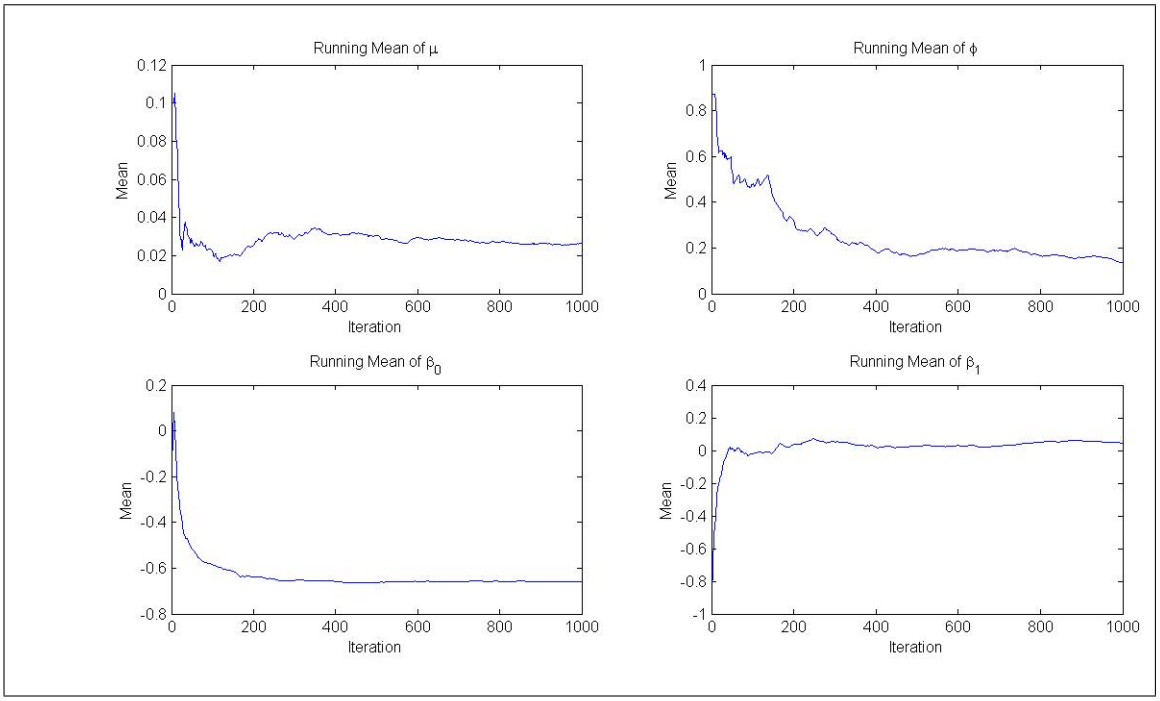

Figure 5.2: Running Mean of Parameters for $2T = 100$, $K = 300$.

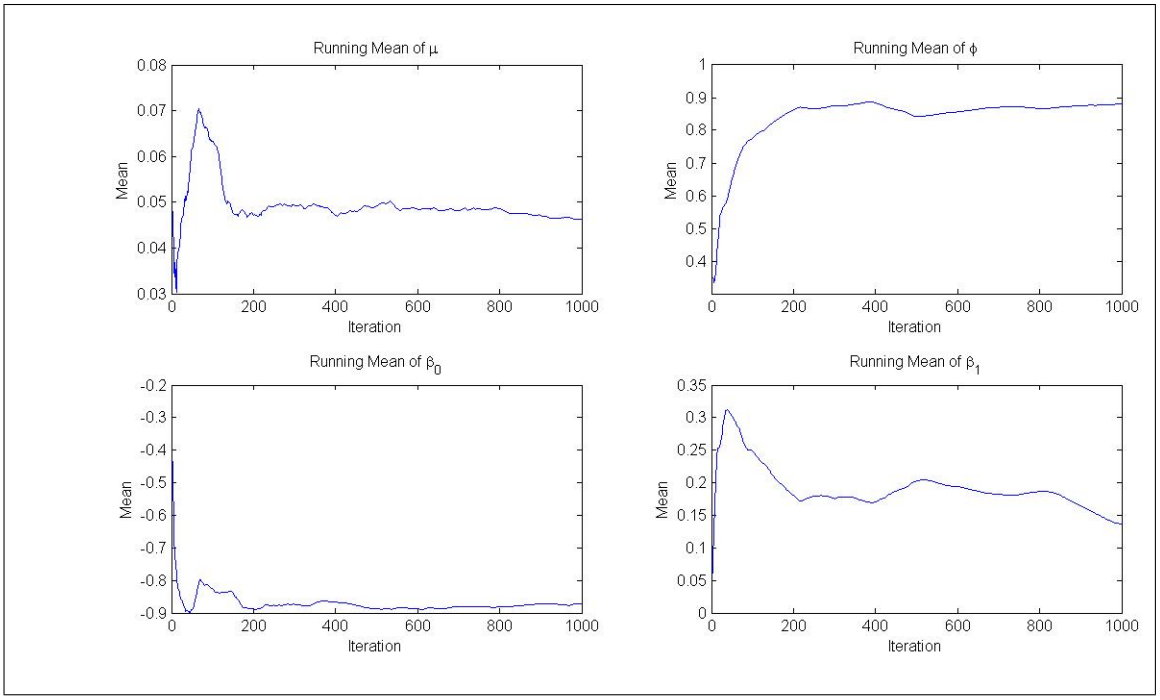

Figure 5.3: Running Mean of Parameters for $2T = 200$, $K = 300$.

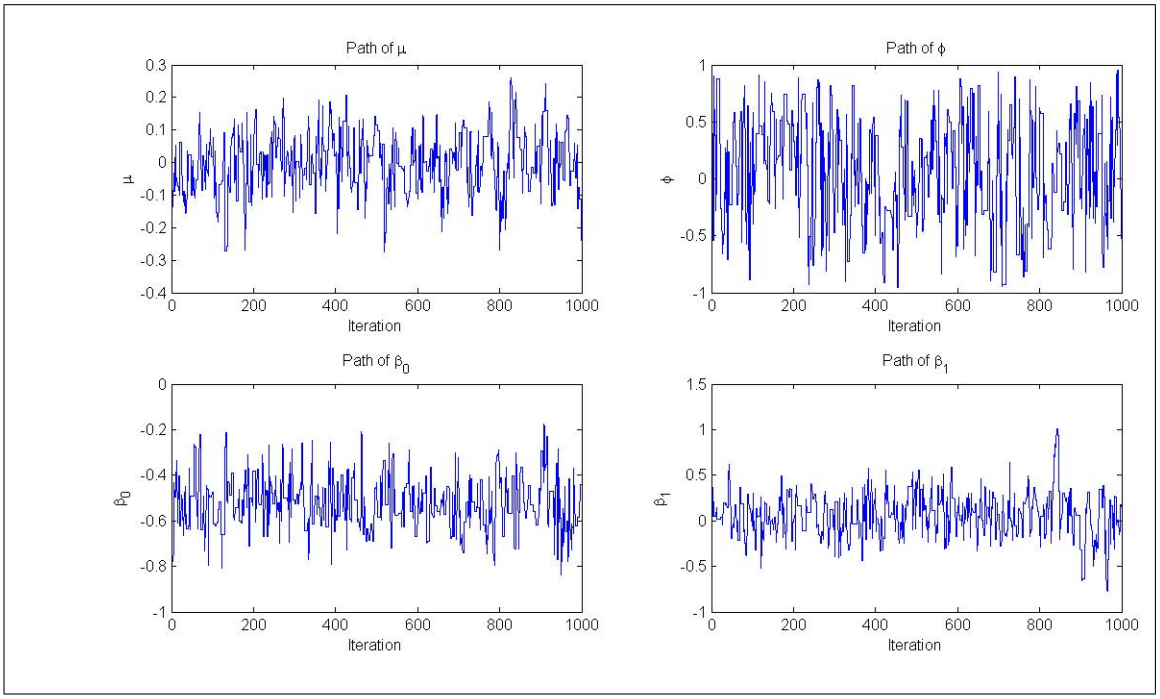

Figure 5.4: Sample Path of Parameters for $2T = 50$, $K = 100$.

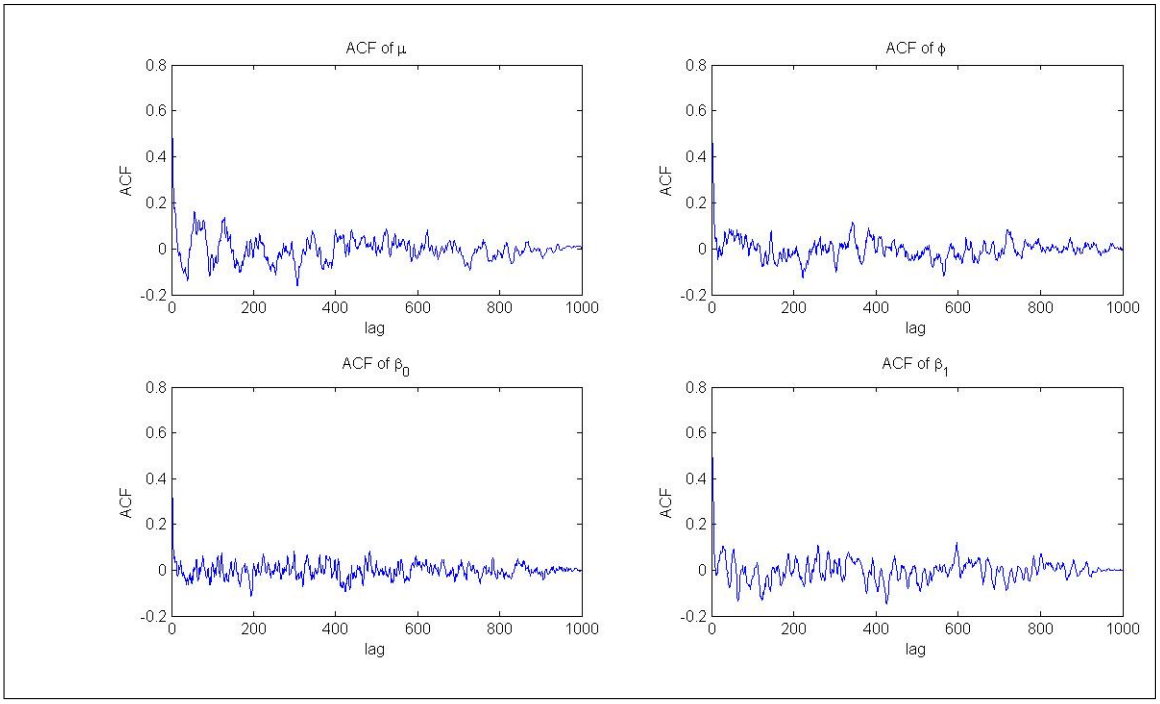

Figure 5.5: ACF Plots for $2T = 50$, $K = 100$.

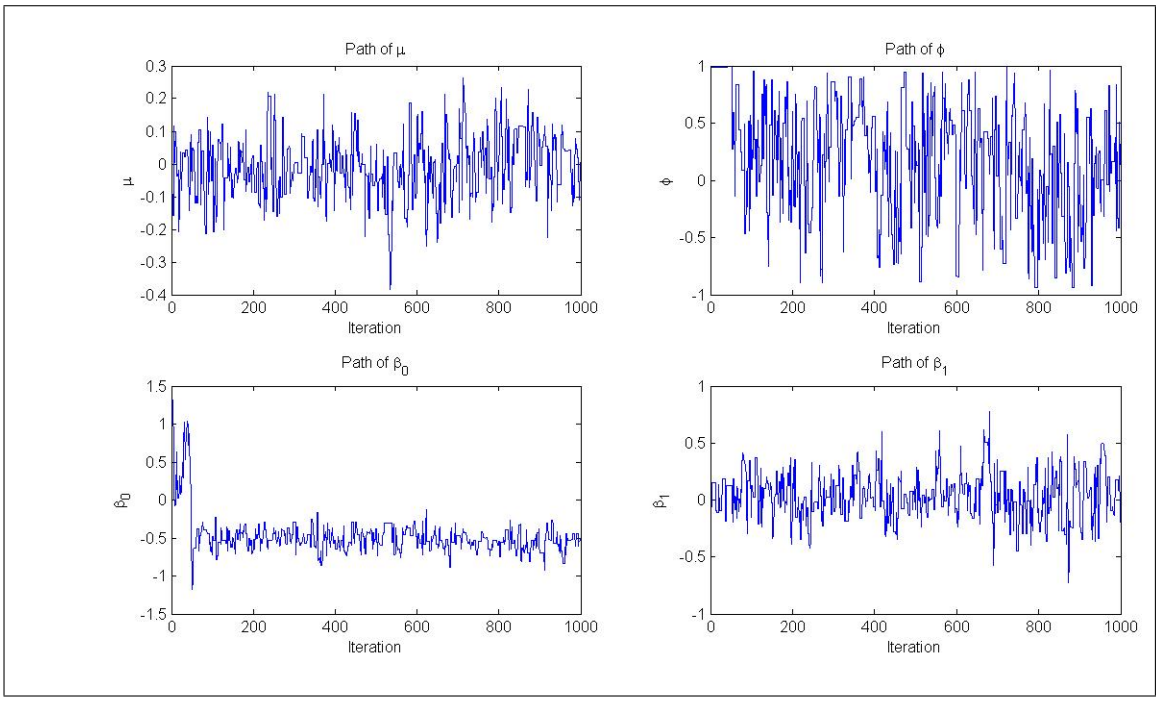

Figure 5.6: Sample Path of Parameters for $2T = 50$, $K = 300$.

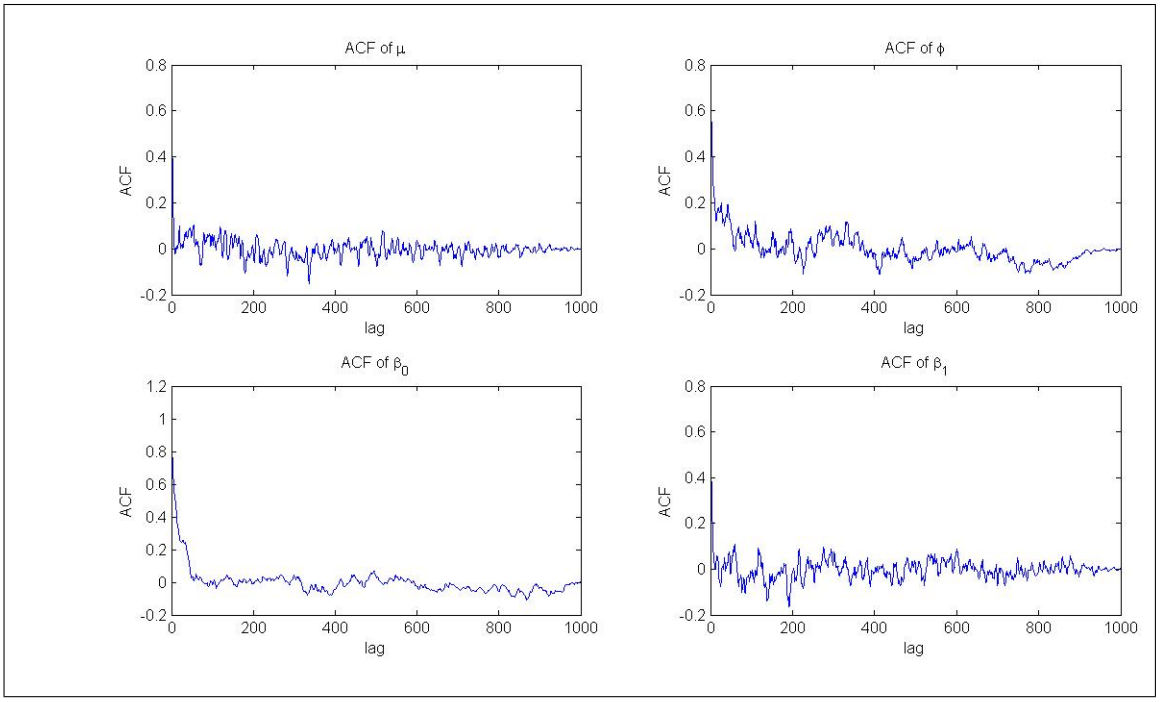

Figure 5.7: ACF Plots for $2T = 50$, $K = 300$.

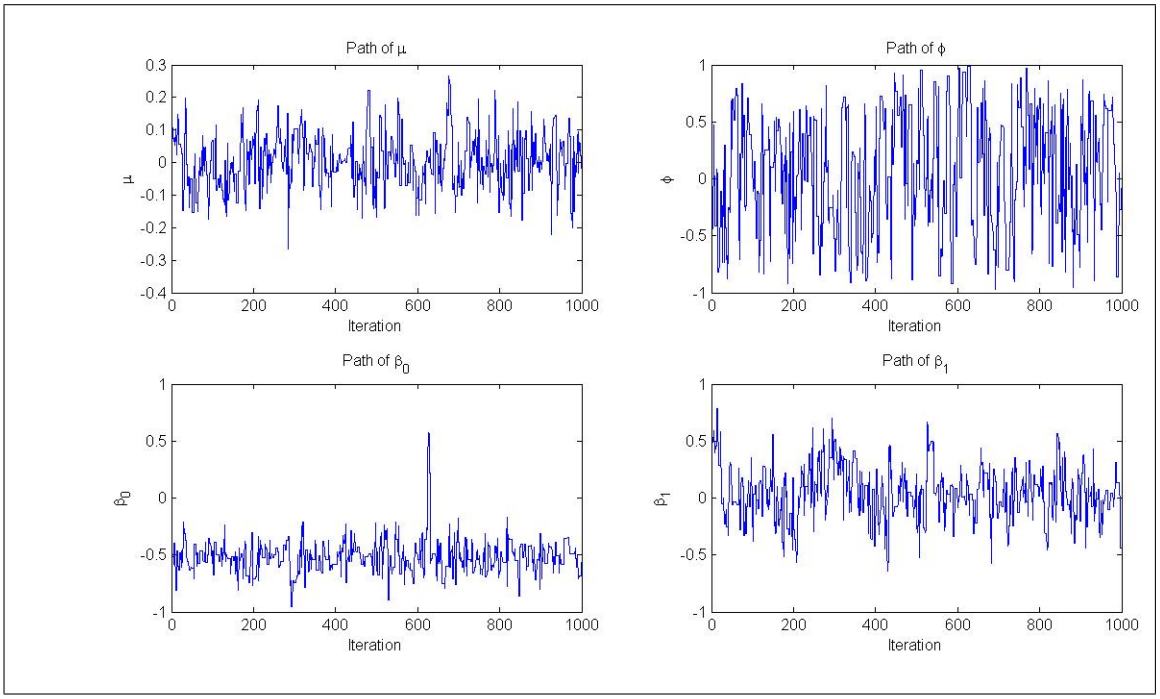

Figure 5.8: Sample Path of Parameters for $2T = 50$, $K = 500$.

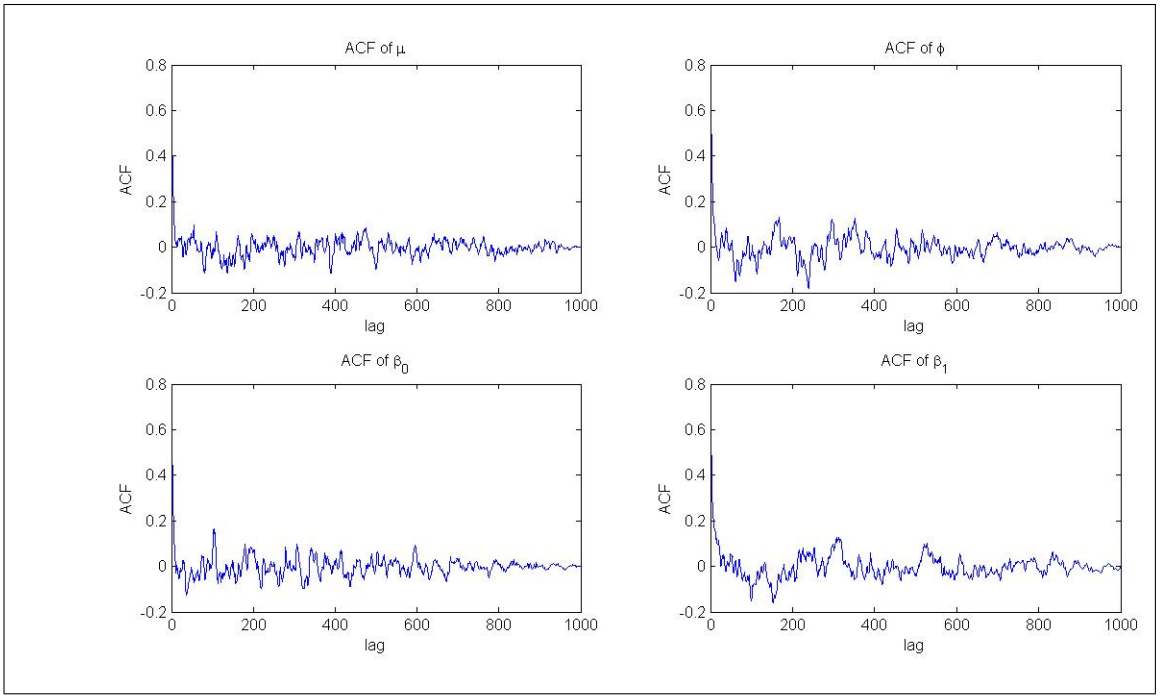

Figure 5.9: ACF Plots for $2T = 50$, $K = 500$.

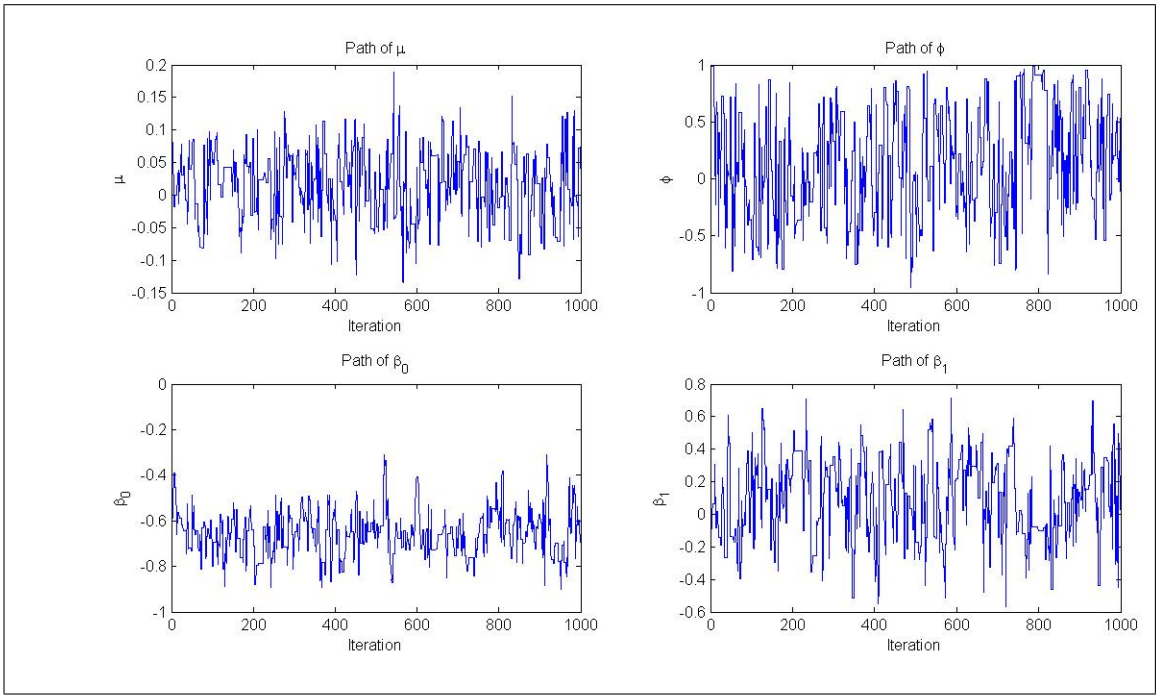

Figure 5.10: Sample Path of Parameters for $2T = 100$, $K = 100$.

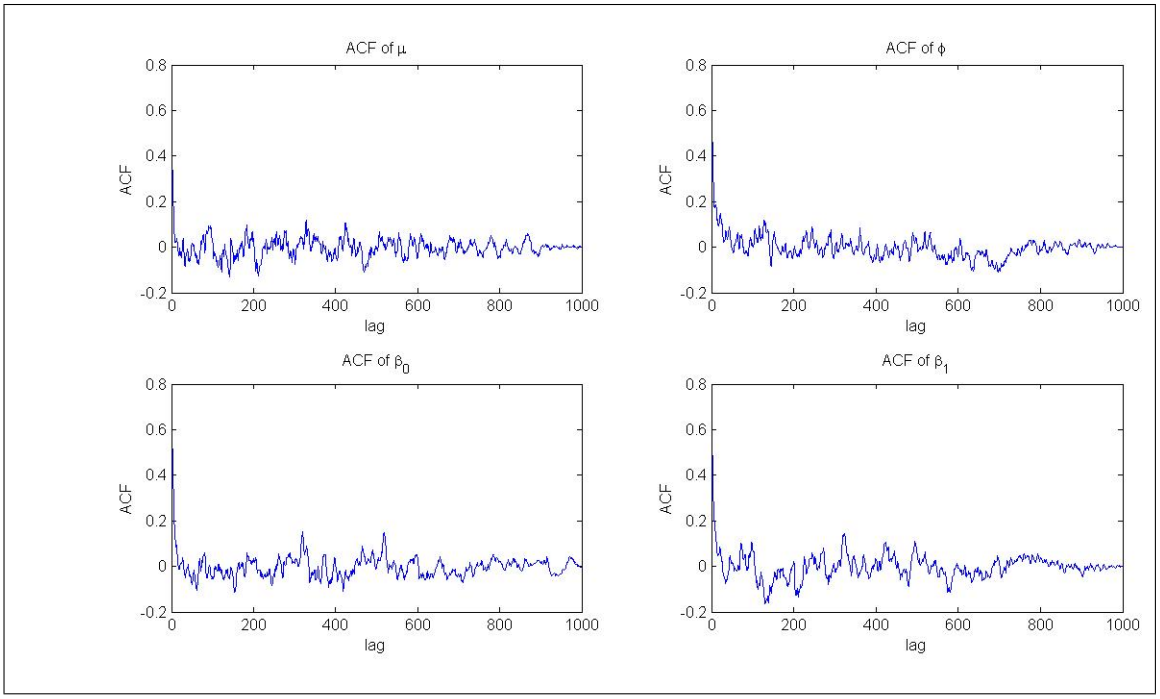

Figure 5.11: ACF Plots for $2T = 100$, $K = 100$.

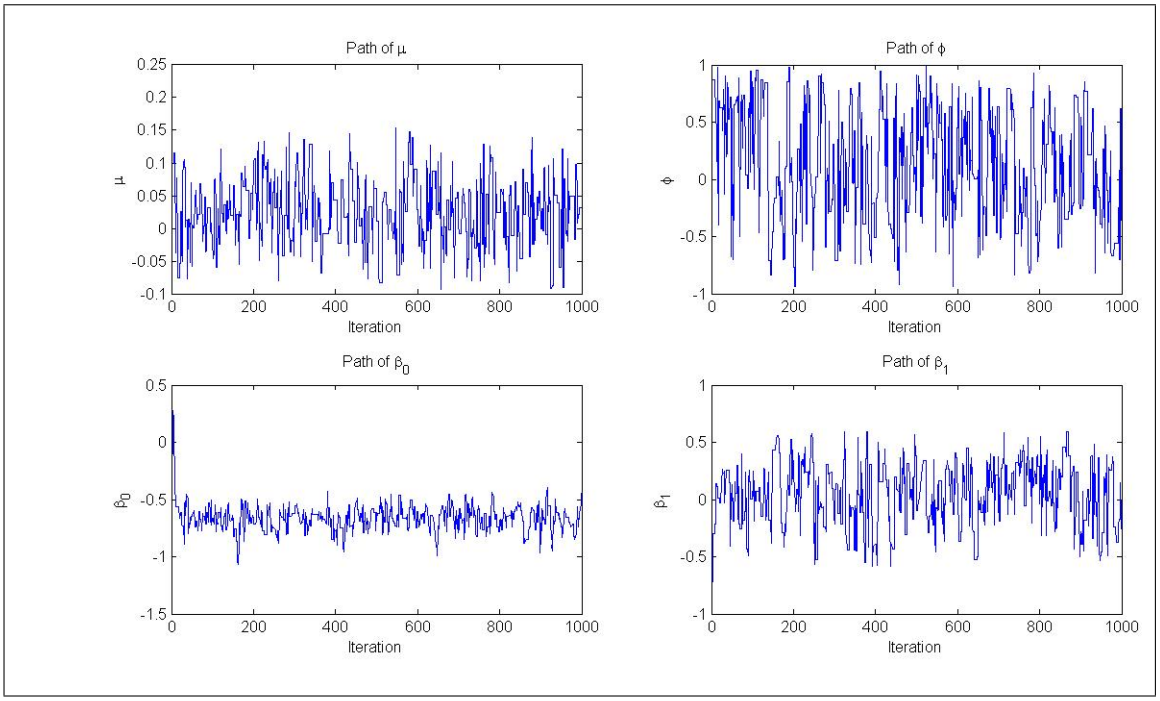

Figure 5.12: Sample Path of Parameters for $2T = 100$, $K = 300$.

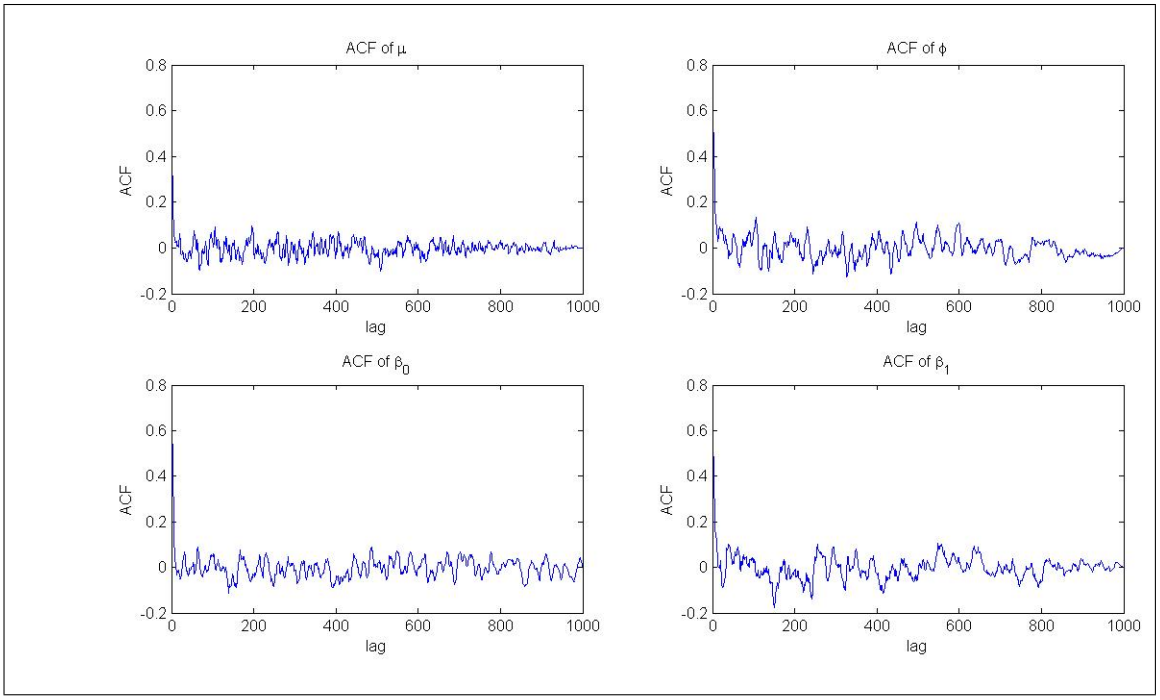

Figure 5.13: ACF Plots for $2T = 100$, $K = 300$.

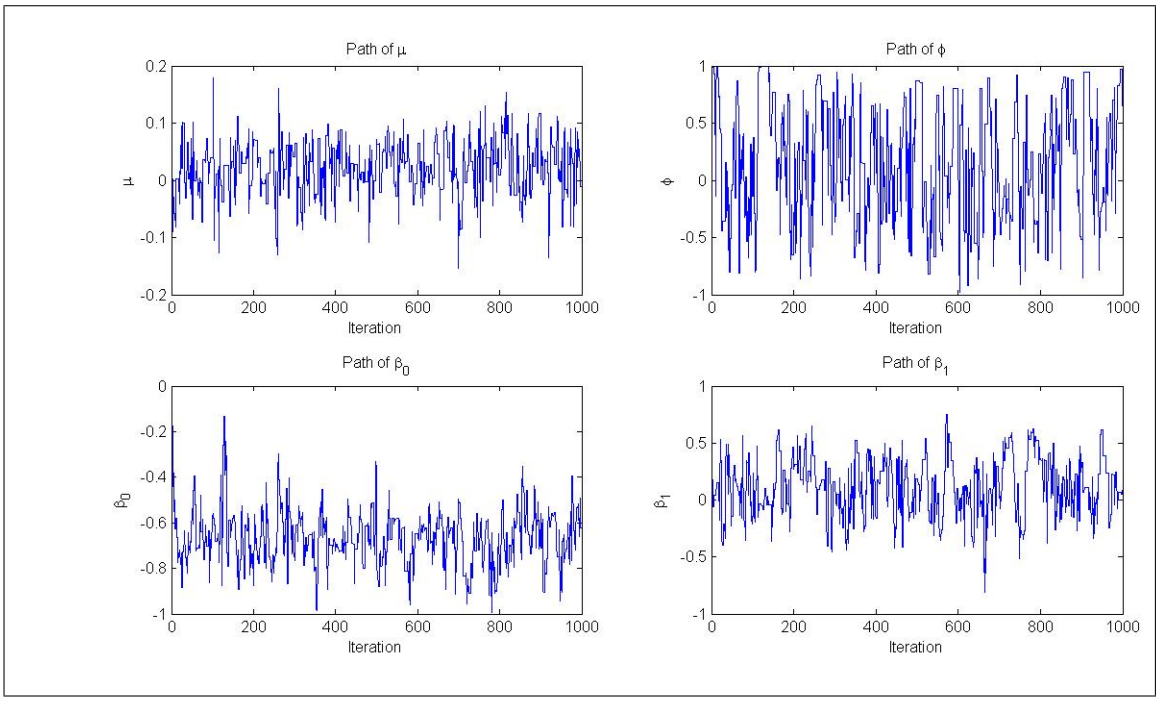

Figure 5.14: Sample Path of Parameters for $2T = 100$, $K = 500$.

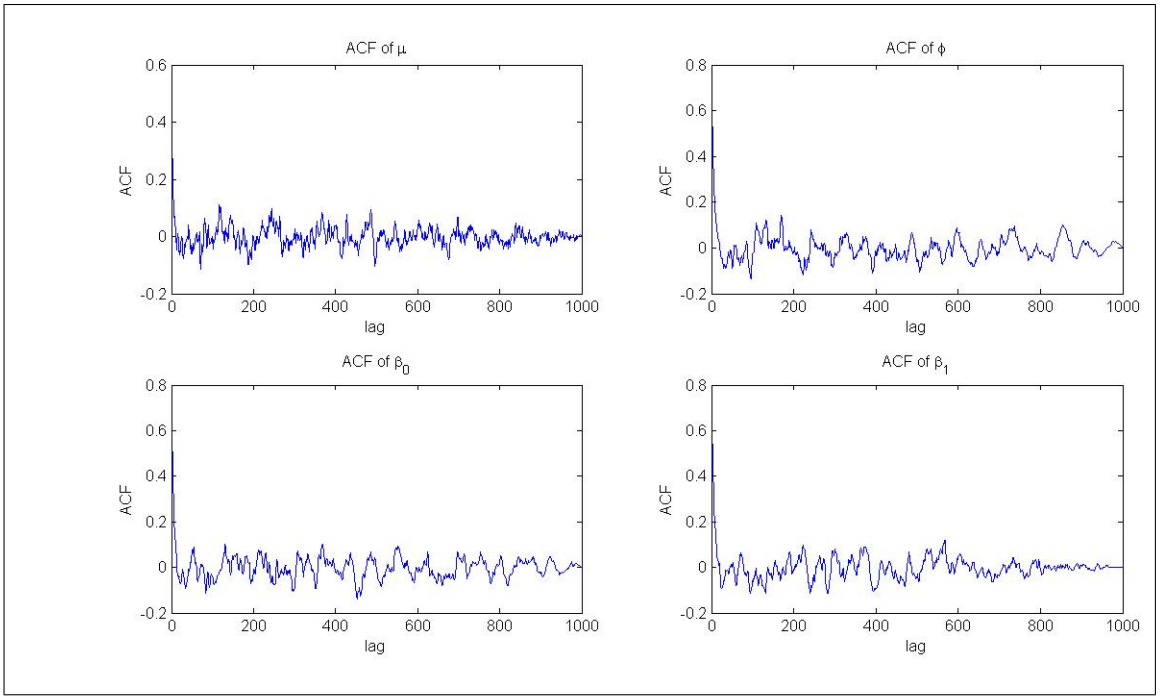

Figure 5.15: ACF Plots for $2T = 100$, $K = 500$.

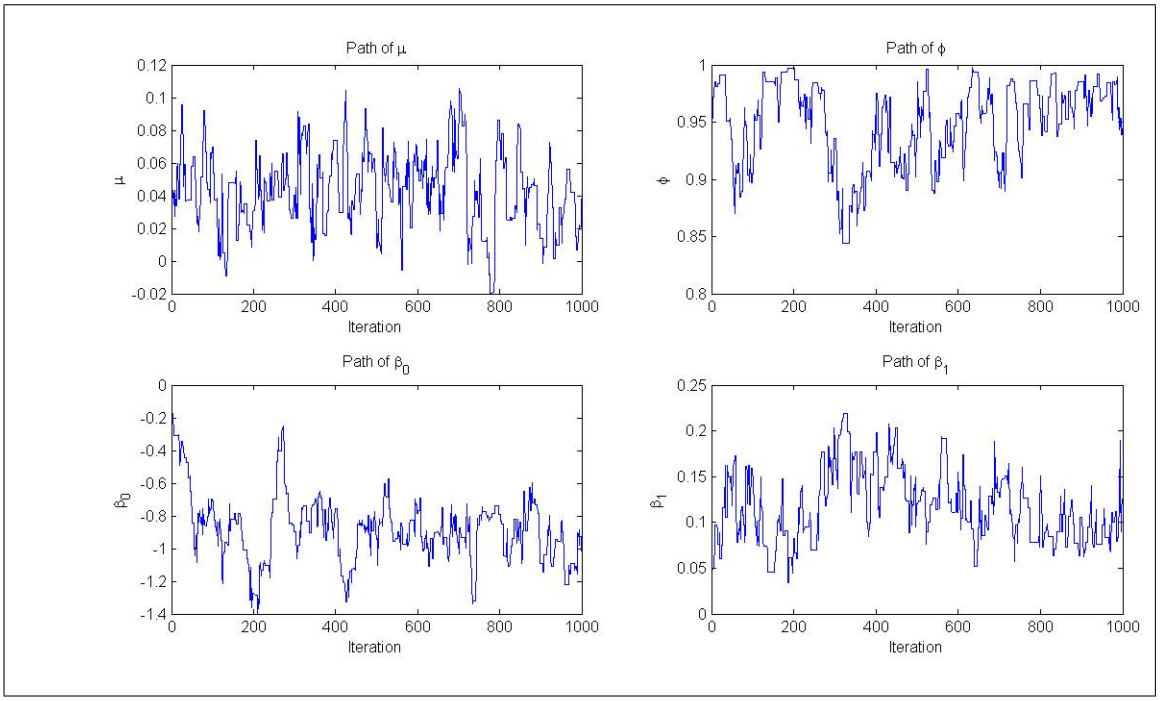

Figure 5.16: Sample Path of Parameters for $2T = 200$, $K = 100$.

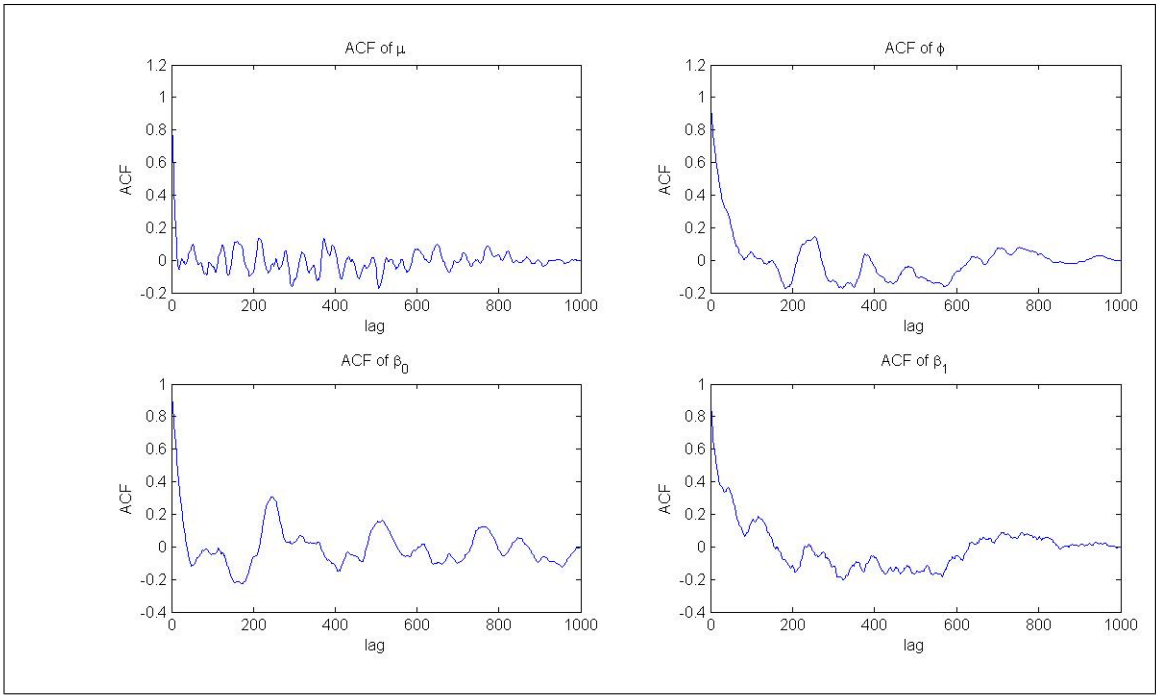

Figure 5.17: ACF Plots for $2T = 200$, $K = 100$.

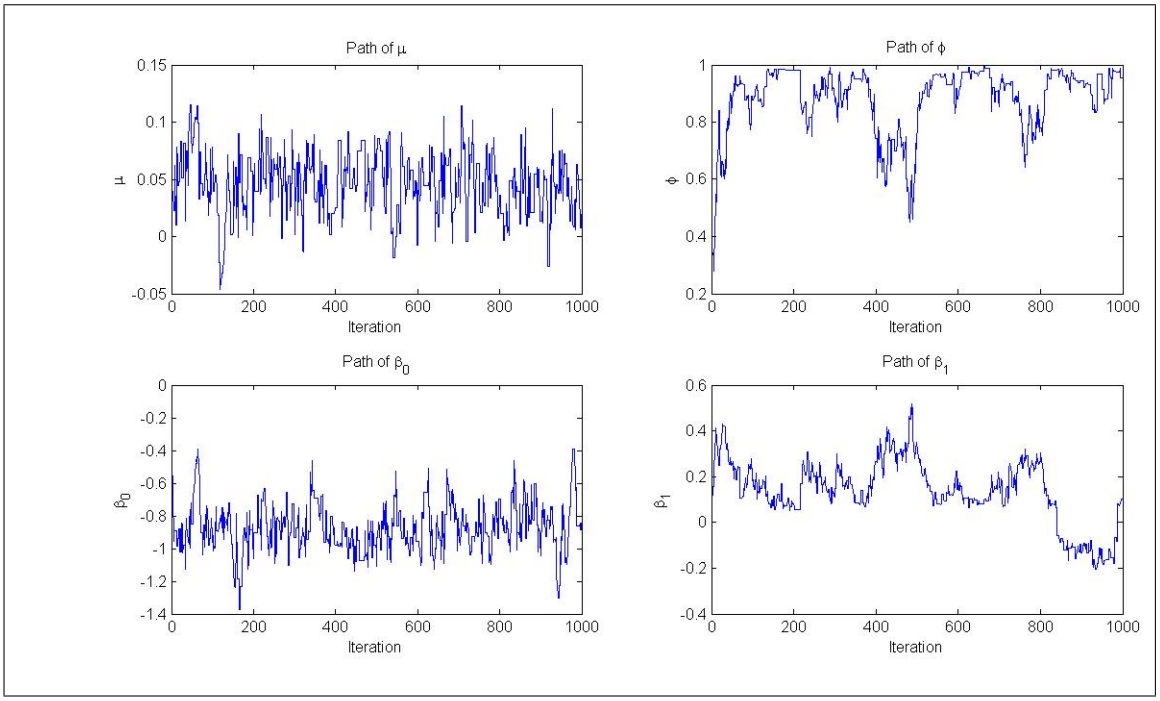

Figure 5.18: Sample Path of Parameters for $2T = 200$, $K = 300$.

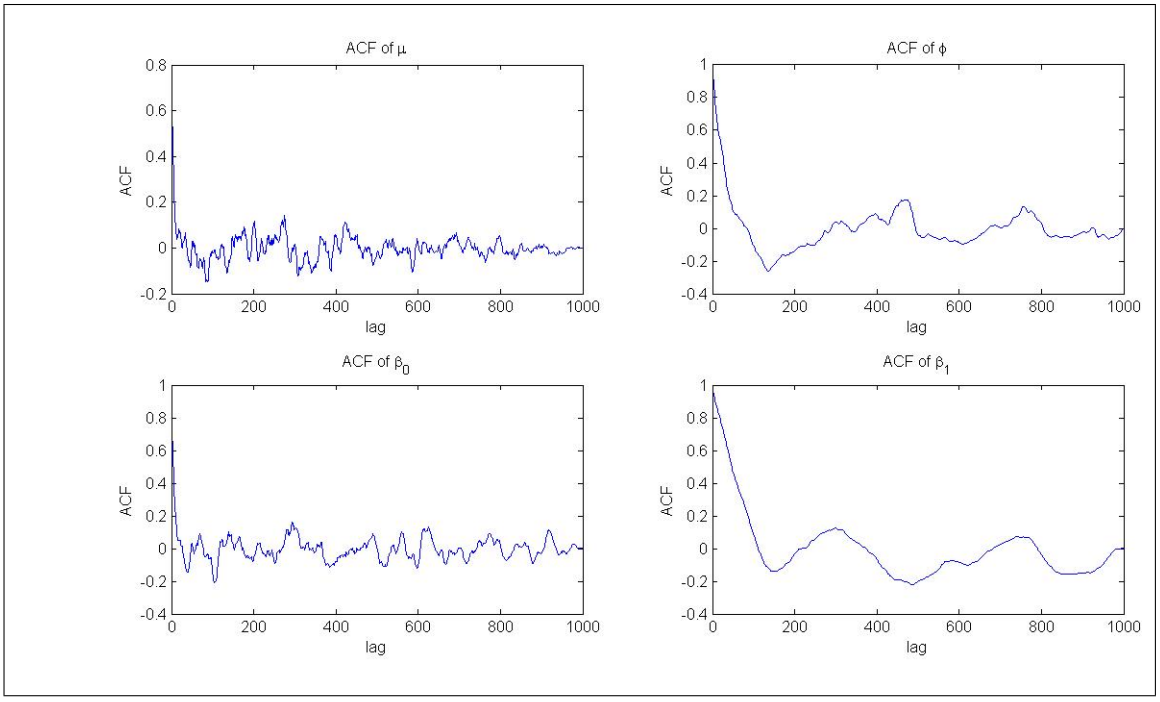

Figure 5.19: ACF Plots for $2T = 200$, $K = 300$.

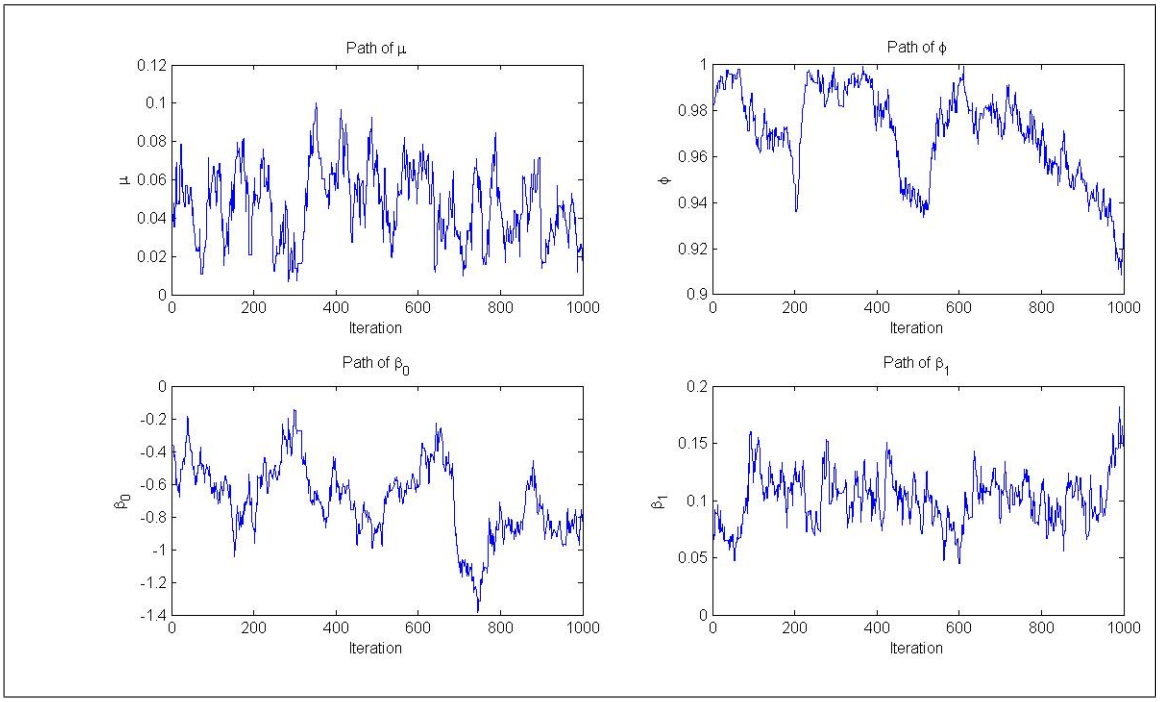

Figure 5.20: Sample Path of Parameters for $2T = 200$, $K = 500$.

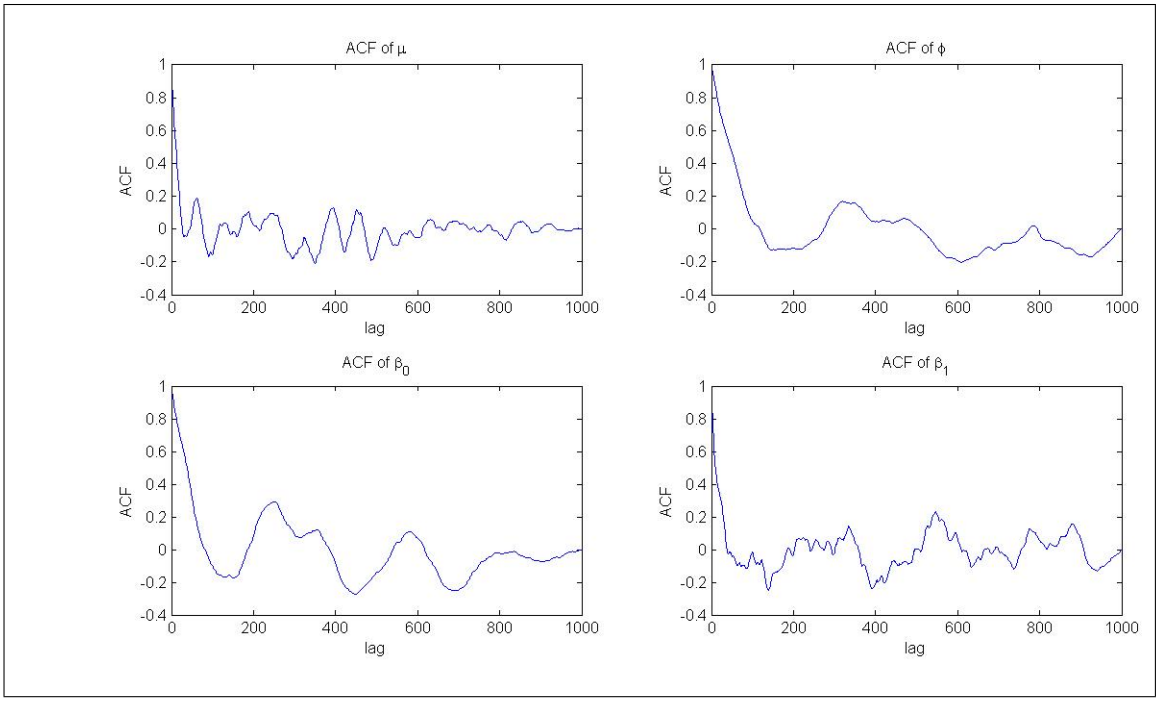

Figure 5.21: ACF Plots for $2T = 200$, $K = 500$.

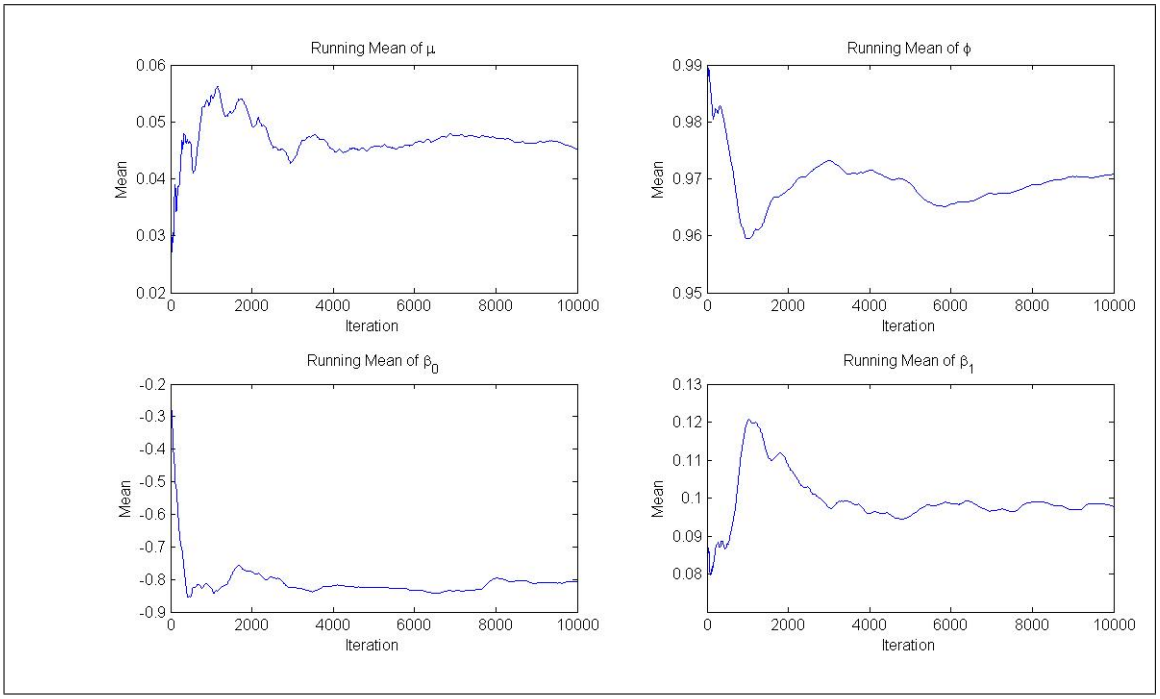

Figure 5.22: Running Mean Plots for $2T = 200$, $K = 300$ and $N = 10000$.

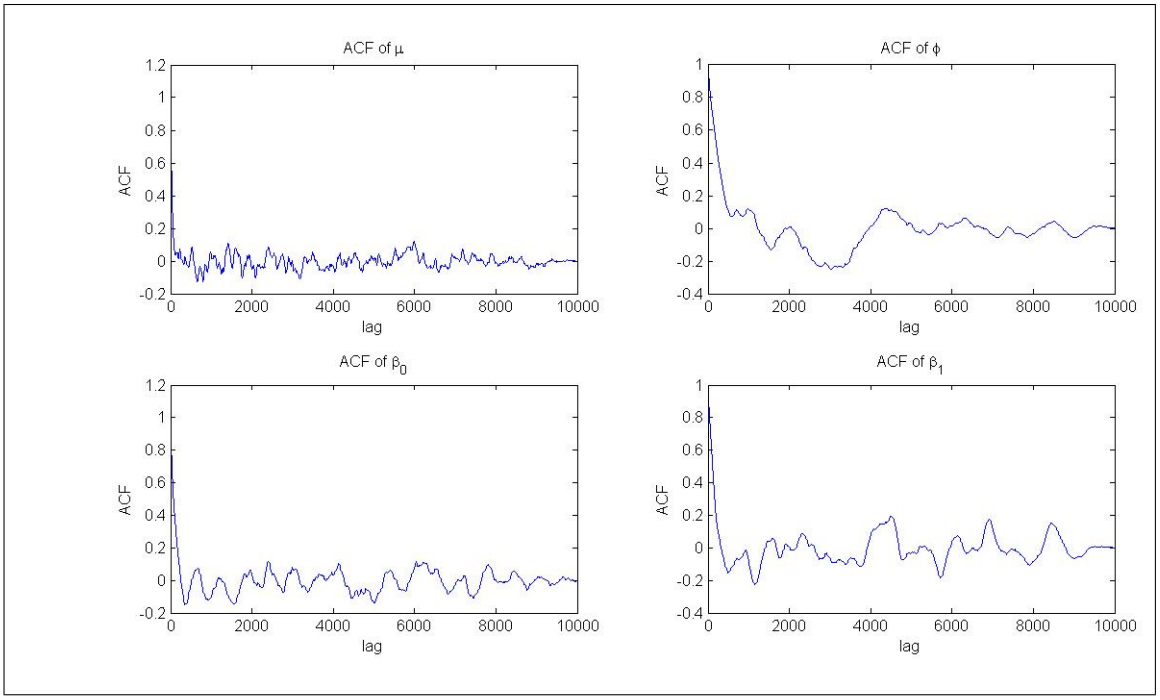

Figure 5.23: ACF Plots for $2T = 200$, $K = 300$ and $N = 10000$.

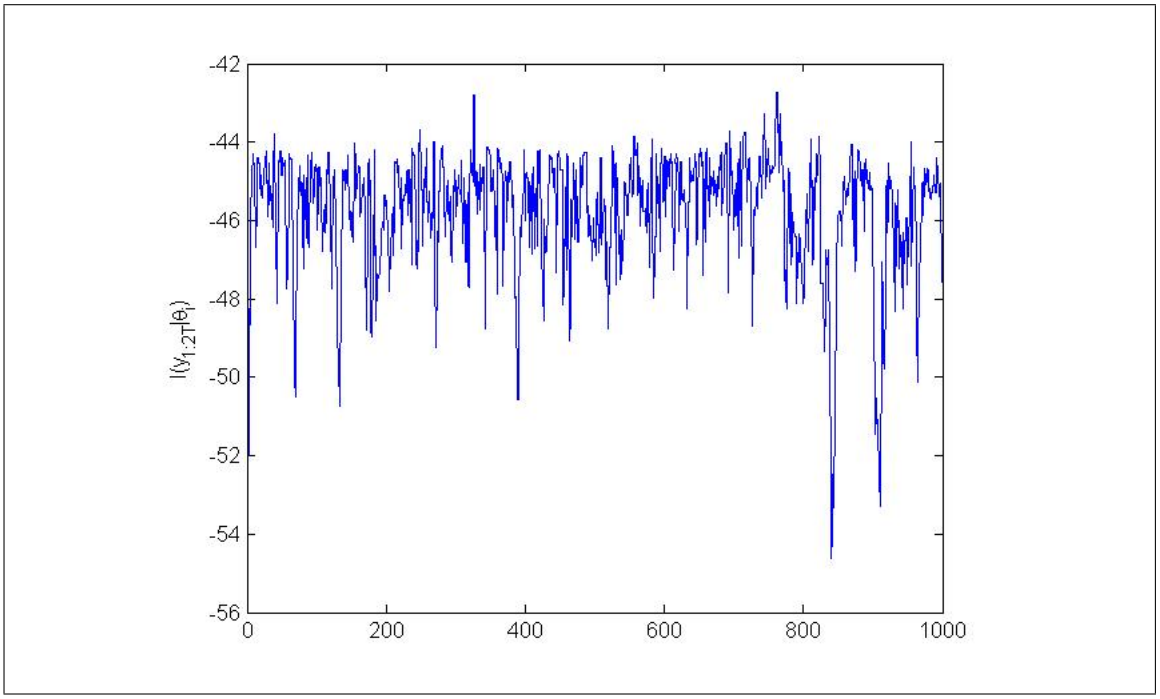

Figure 5.24: Log Likelihood for $2T = 50$, $K = 100$.

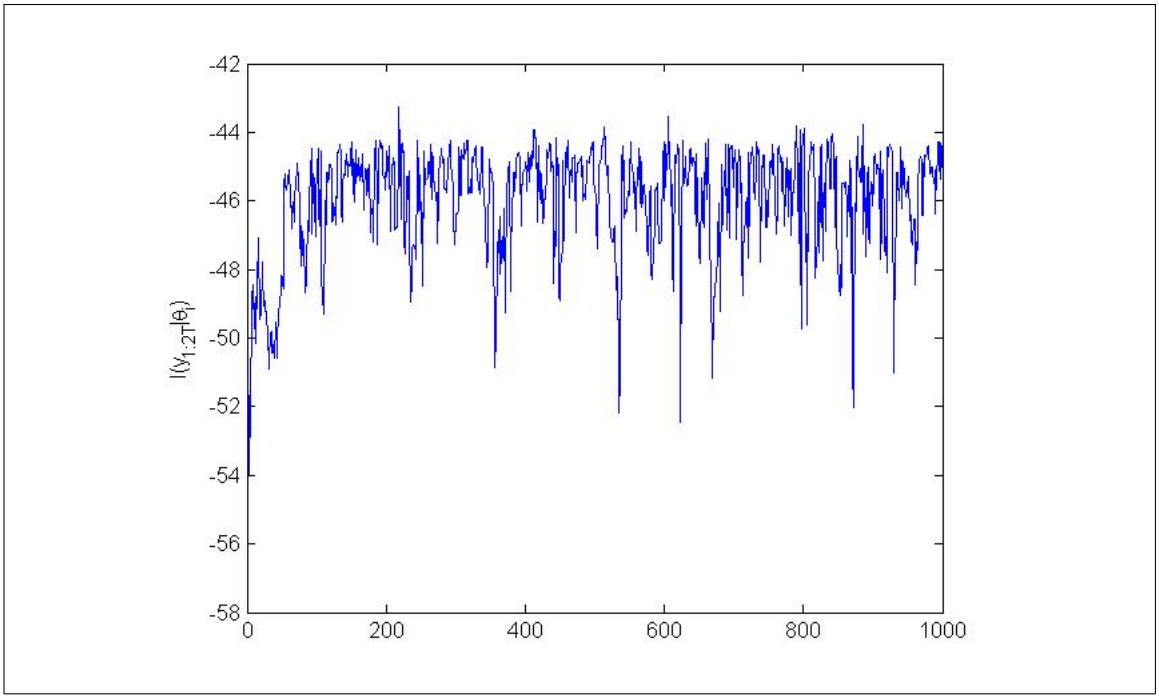

Figure 5.25: Log Likelihood for $2T = 50$, $K = 300$.

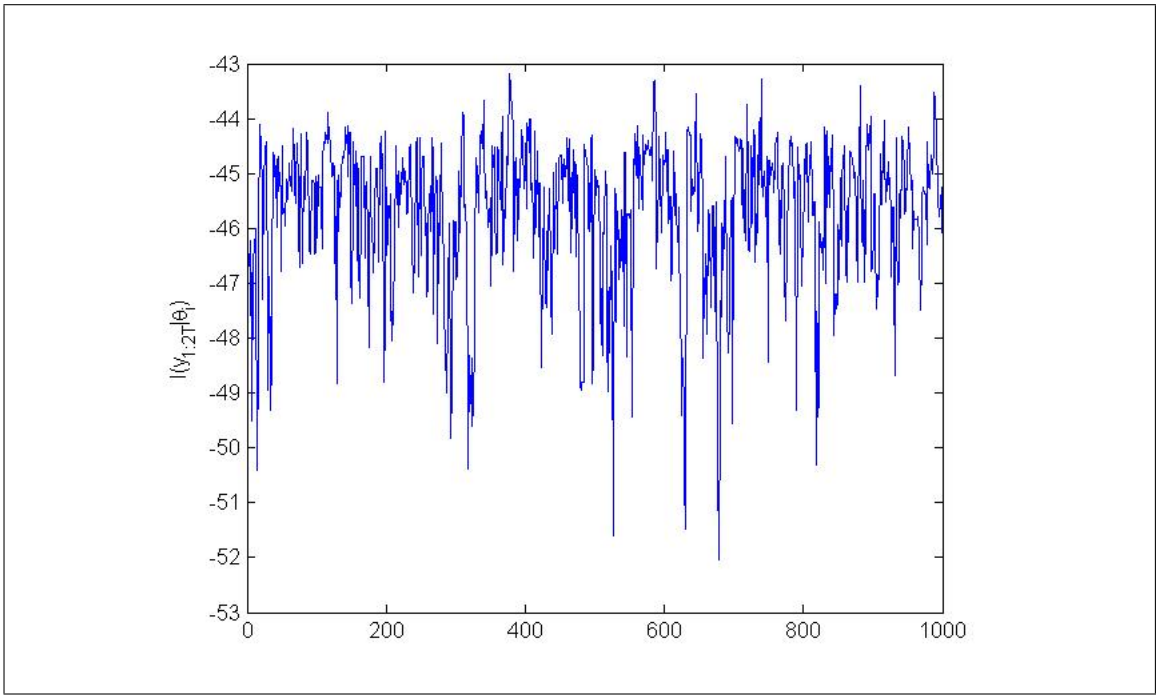

Figure 5.26: Log Likelihood for $2T = 50$, $K = 500$.

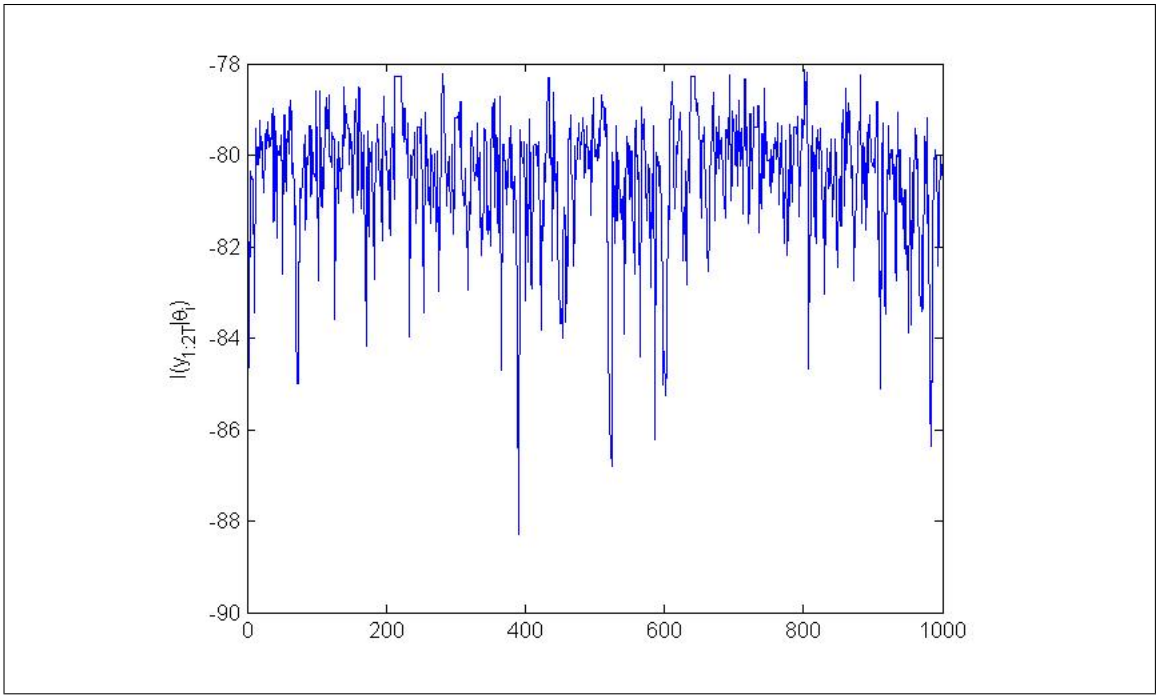

Figure 5.27: Log Likelihood for $2T = 100$, $K = 100$.

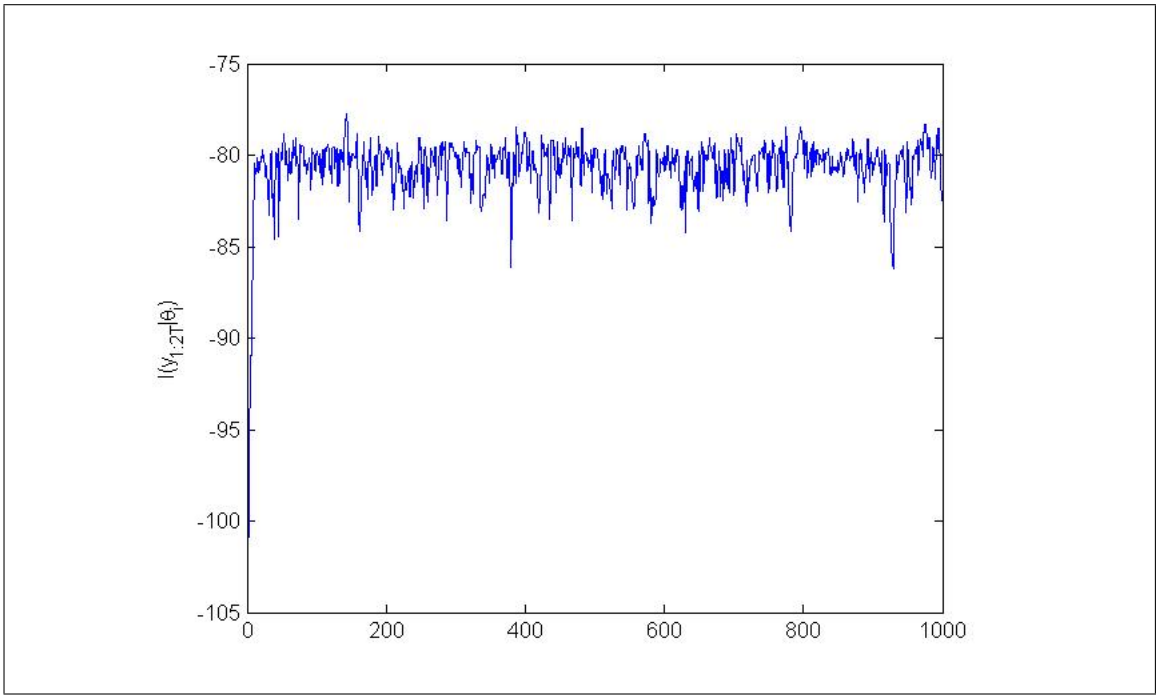

Figure 5.28: Log Likelihood for $2T = 100$, $K = 300$.

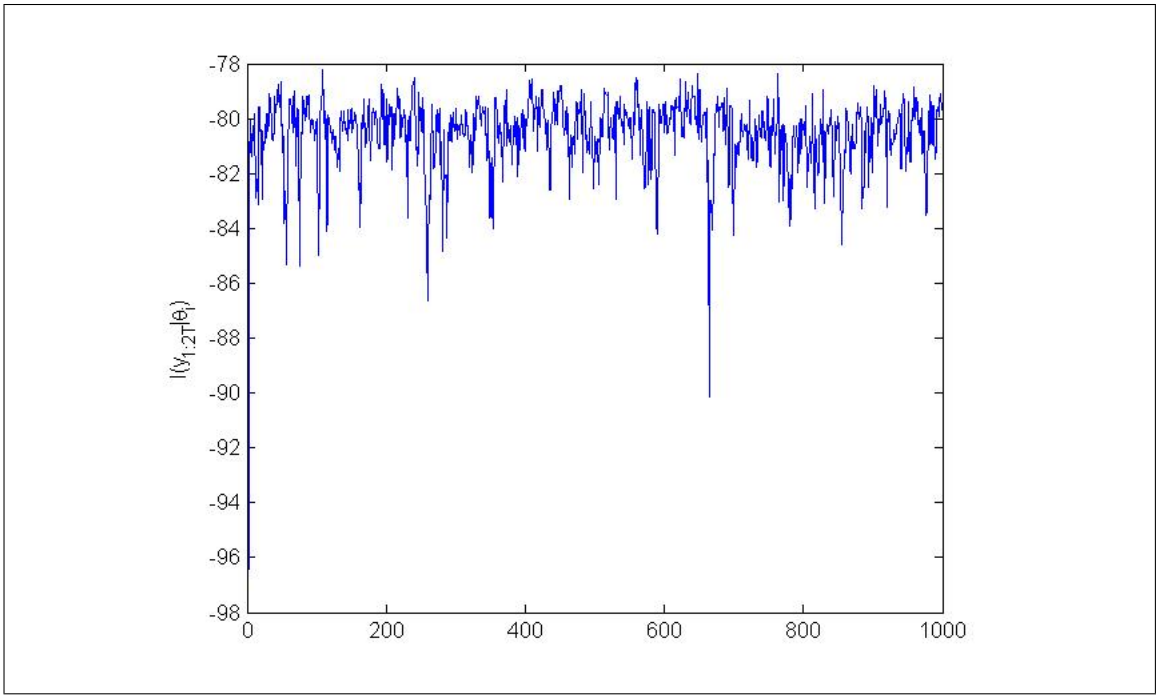

Figure 5.29: Log Likelihood for $2T = 100$, $K = 500$.

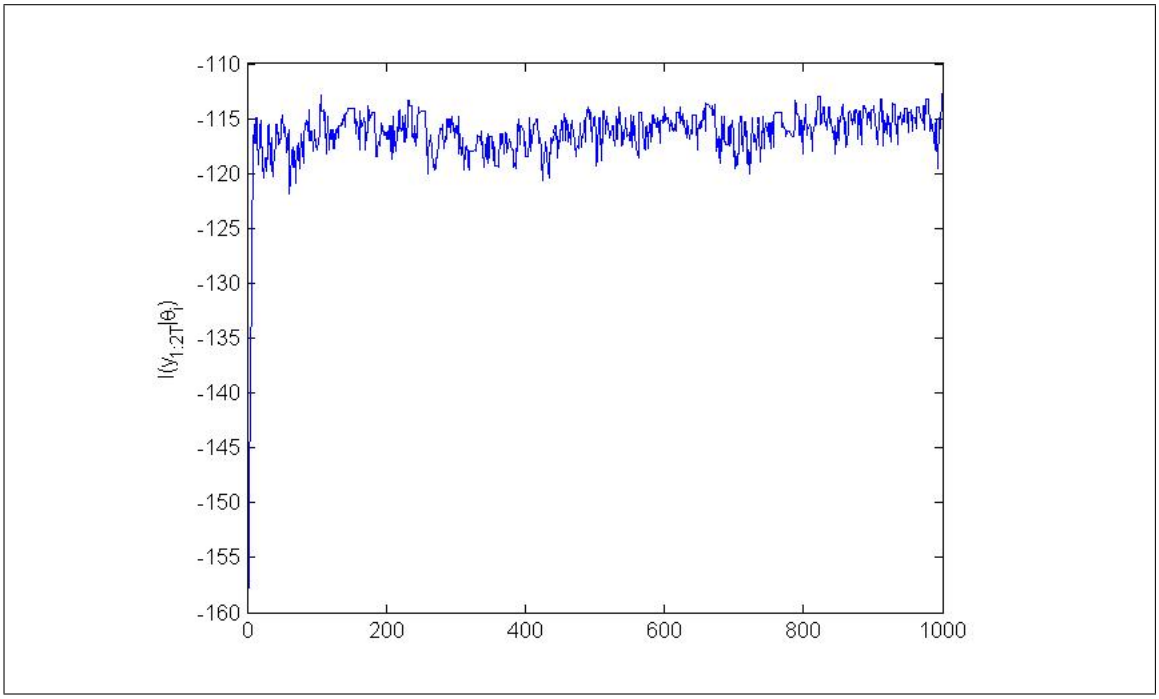

Figure 5.30: Log Likelihood for $2T = 200$, $K = 100$.

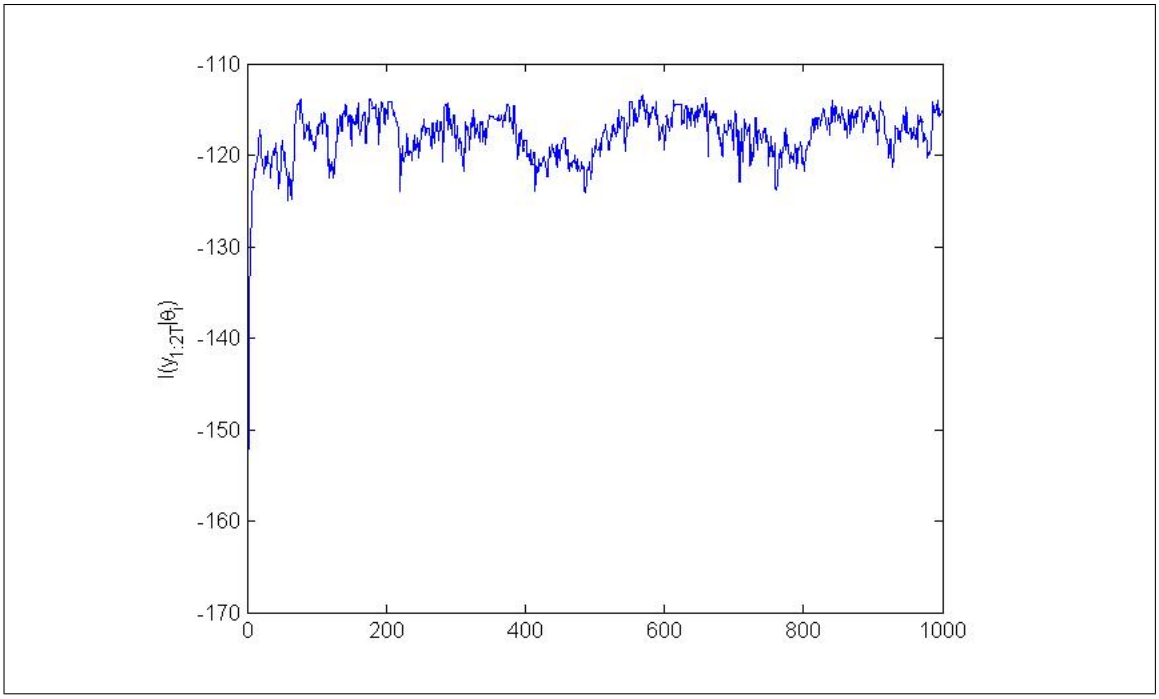

Figure 5.31: Log Likelihood for $2T = 200$, $K = 300$.

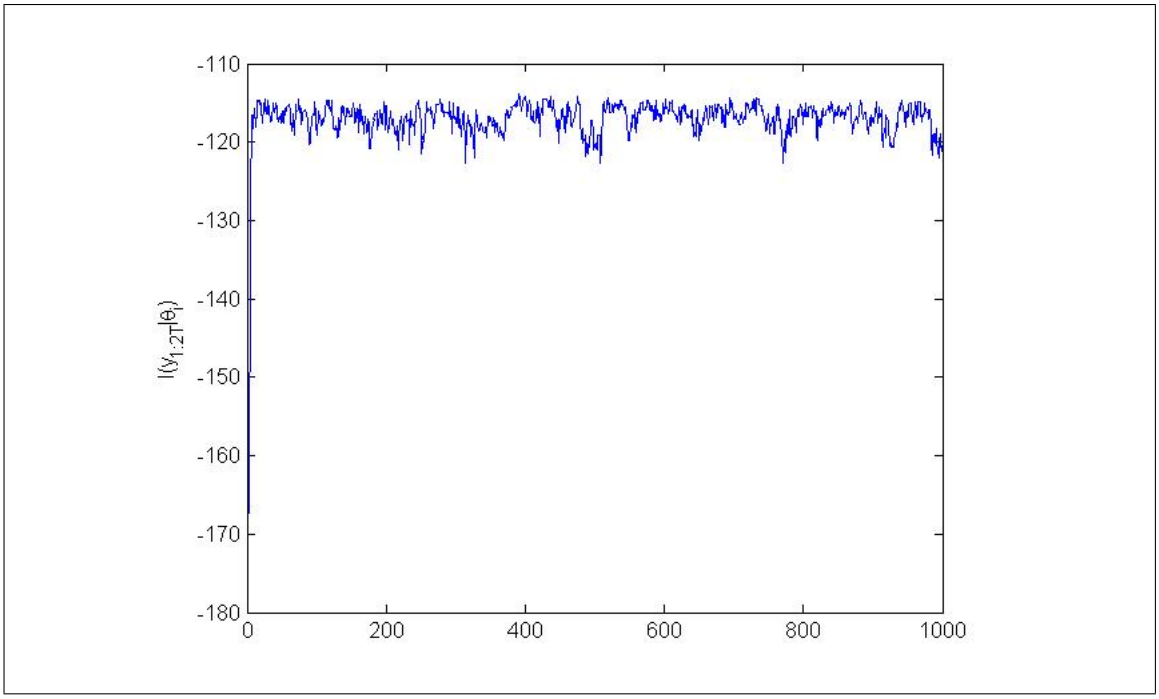

Figure 5.32: Log Likelihood for $2T = 200$, $K = 500$.

Chapter 6

Conclusion

In this thesis, we reviewed the methodology of particle filters and various Markov chain Monte Carlo methods for making inferences for the hidden Markov models in Chapter 2. In particular, we touched on the use of particle Markov chain Monte Carlo methods and introduced substitution algorithm in Bayesian inferences of a hidden Markov model in this chapter. In Chapter 3, we introduced the proposed algorithm formally for making use of parallel particle filters and the associated framework. The unbiasedness property of the proposed estimates using parallel particle filters are proven, motivated by the idea used by Chan and Lai (2013). Simulations are used to illustrate the efficiency of the parallel particle filters estimate for a linear Gaussian hidden Markov model in Chapter 4. Finally, a numerical study of real data using a discrete time Gaussian stochastic volatility model was done in Chapter 5. The parallel particle filter algorithm is used in the particle Markov chain Monte Carlo algorithm to obtain approximations of the posterior distribution of the model's parameters. The performance of implementing the parallel particle filter in the PMCMC algorithm is assessed in this chapter.

In Chapter 3, we introduced two martingale difference expressions for the proof of unbiasedness property in Theorems 3.4 and 3.6. While these are generalisation for the estimates given by Chan and Lai (2013), the proofs have to be extended to cover the use of data segmentation. As remarked in the chapter, one form can be used to

prove a central limit theorem for the proposed estimates and another can be used to provide approximations to the standard error of the estimates. The proofs of these are given by Chan *et al.* (2014), which are extensions of the proofs given by Chan and Lai (2013).

We used parallel particle filters for the estimation of smoothed means in our numerical study in Chapter 4. Future work could be done in improving the smoothing techniques by marrying existing methodology with our proposed parallel particle filter algorithm.

Throughout this thesis, we split the sequence into disjoint subsequences of equal length. We shall remark that the subsequences need not be of equal length. Further, the requirement that the subsequences are disjoint can be relaxed. It could be advantageous to include additional observation at the edges of the subsequence to smooth out the joining of the sample path.

In Chapter 4, we made a remark on the number of subsequences used for the parallel particle filter algorithm. We indicated the optimum number of subsequences used for the given hidden Markov model when the parameters are fixed. We proposed that the optimal number of subsequences could be work for future research which is of practical interest in the efficient implementation of the parallel particle filter algorithm.

Furthermore, we have touched on the use of subsampling to achieve $\mathcal{O}(K)$ computational cost where K is the number of particles used for the particle filter. In our numerical study, we used a discrete uniform distribution to perform the subsampling technique. Future research can look into the optimal selection of subsampling distribution to achieve estimates with better performance and efficiency when compared with the case where subsampling is not used.

As mentioned in Chapter 3, our proposed method is different in principle from the existing methodologies that harness parallel computing. Future work could be done to incorporate our method into these methodologies to achieve additional computa-

tional cost savings while retaining the advantages of these methodologies.

While we have run simulations with real data using the parallel particle filters algorithm, we have only make use of the particle Markov chain Monte Carlo algorithm for our numerical study. One might be interested to implement the parallel particle filter algorithm in the substitution algorithm or the SMC^2 algorithm to ascertain the validity and advantages, if any. One could also investigate the efficiency of the parallel particle filter for the optimal number of particles to be used.

As a final remark, one might be interested to implement the parallel particle filter algorithm for higher dimensional problems. Future research could be done on harnessing the use of parallel particle filters for cases where unbiased estimates are needed in the implementation of such algorithms.

Bibliography

[1] Andrieu, C., Doucet, A., and Holenstein, R. Particle Markov chain Monte Carlo methods (with discussion). *Journal of the Royal Statistical Society B 72*, 3 (2010), 269–342.

[2] Andrieu, C., and Gareth O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics 37*, 2 (2009), 697–725.

[3] Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Proceedings on Signal Processing 50*, 2 (2002), 174–188.

[4] Baker, J. E. Adaptive selection methods for generic algorithms. *In Proceedings of the International Conference on Genetic Algorithms and Their Applications* (1985), 101–111.

[5] Baker, J. E. Reducing bias and inefficiency in the selection algorithm. *In Proceedings of the Second International Conference on Genetic Algorithms and Their Applications* (1987), 14–21.

[6] Barndorff-Nielsen, O. E., and Shephard, N. Econometric analysis of realized volatility and its use in estimating stochastic volatility models. *Journal of the Royal Statistical Society B 64*, 2 (2002), 253–280.

[7] Beaumont, M. A. Estimation of population growth or decline in genetically monitored population. *Genetics 164*, 3 (2003), 1139–1160.

[8] Beaumont, M. A., Cornuet, J.-M., Marin, J.-M., and Robert, C. P. Adaptive approximate Bayesian computation. *Biometrika 96*, 4 (2009), 983–990.

[9] Billingsley, P. *Probability and Measure 3rd Edition*. Wiley, 1995.

[10] Briers, M., Doucet, A., and Singh, S. S. Sequential auxiliary particle belief propagation. *2005 7th International Conference on Information Fusion (FUSION)*.

[11] Bunke, H., and Caelli, T. *Hidden Markov Models: Applications in Computer Vision*. World Scientific, 2001.

[12] Cappé, O., Moulines, E., and Rydén, T. *Inference in hidden Markov models*. Springer: New York, 2005.

[13] Chan, H. P., Jasra, A., and Heng, C. W. Theory of parallel particle filters for hidden Markov models. *Manuscript* (2014).

[14] Chan, H. P., and Lai, T. L. A general theory of particle filters in hidden Markov models and some applications. *The Annals of Statistics 41*, 6 (2013), 2877–2904.

[15] Chan, H. P., and Lai, T. L. A new approach to Markov chain Monte Carlo with applications to adaptive particle filters. *Manuscript* (2014).

[16] Chib, S., Nardari, F., and Shephard, N. Markov chain Monte Carlo methods for stochastic volatility models. *Journal of Econometrics 108*, 2 (2002), 281–316.

[17] Chopin, N. A sequential particle filter for static models. *Biometrika 89* (2002), 539–552.

[18] Chopin, N., Jacob, P. E., and Papaspiliopoulos, O. SMC^2: an efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society B 75*, 3 (2013), 397–426.

[19] Chung, K. L. *A Course in Probability Theory 3rd Edition*. Academic Press, 2001.

[20] Creal, D. A survey of sequential Monte Carlo methods for economics and finance. *Econometric Reviews 31* (2012), 245–296.

[21] Del Moral, P. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer: New York, 2004.

[22] Del Moral, P., Doucet, A., and Jasra, A. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society B 68* (2006), 411–436.

[23] Douc, R., Cappé, O., and Moulines, E. Comparison of resampling schemes for particle filtering. *In Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis (ISPA)* (2005).

[24] Doucet, A., de Freitas, N., and Gordon, N. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2001.

[25] Doucet, A., and Johansen, A. M. A tutorial on particle filtering and smoothing: fifteen years later. *Technical report (Department of Statistics, University of British Columbia)* (2008).

[26] Doucet, A., Pitt, M., and Deligiannidis, G. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *Unpublished paper* (2014).

[27] Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[28] Durrett, R. *Probability: Theory and Examples 2nd Edition.* Duxbury, 1995.

[29] Efron, B. Bootstrap methods: another look at the Jacknife. *The Annals of Statistics 7*, 1 (1979), 1–26.

[30] Efron, B., and Tibshirani, R. J. *An Introduction to the Bootstrap.* Chapman & Hall, 1993.

[31] Elliott, R. J. New finite dimensional filters and smoothers for Markov chains observed in Gaussian noise. *IEEE Trans. Signal Process 39* (1993), 265–271.

[32] Fearnhend, P., Wyncoll, D., and Tawn, J. A sequential smoothing algorithm with linear computational cost. *Biometrika 97*, 2 (2010), 447–464.

[33] Flury, T., and Shephard, N. Bayesian inference based only on simulated likelihood: particle filter analysis of dynamic economic models. *Econometric Theory 27*, 5 (2011), 933–956.

[34] Geweke, J. Evaluating the accuracy of sampling based approaches to the calculation of posterior moments. In *Bayesian Statistics 4, Bernardo, J. M., Berger, J. O., Dawid, A. P. and Smith, A. F. M. (eds.)* (1992), Oxford University Press, pp. 169–193.

[35] Givens, G. H., and Hoeting, J. A. *Computational Statistics.* Wiley, 2005.

[36] Gordon, N., Salmond, D., and Smith, A. F. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc. F, Radar Signal Process 140* (1993), 107–113.

[37] Hamilton, J. D. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica 57* (1994), 357–384.

[38] Hautsch, N., and Ou, Y. Discrete-time stochastic volatility models and MCMC-based statistical inference. *SFB 649 Discussion Paper 2008-063* (2008).

[39] J. Carpenter, P. C., and Fearnhead, P. An improved particle filter for non-linear problems. *IEE Proc., Radar Sonar Navigation 146* (1999), 2–7.

[40] Jelinek, F. *Statistical Methods for Speech Recognition*. MIT Press, 1997.

[41] Kalman, R. E., and Bucy, R. New results in linear filtering and prediction theory. *J. Basic Eng., Trans. ASME, Series D 83* (1961), 95–108.

[42] Kim, C., and Nelson, C. *State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications*. MIT Press, 1999.

[43] Kitagawa, G. Monte Carlo filter and smoother for non-Gaussian non-linear state space models. *Journal of Computational and Graphical Statistics 5* (1996), 1–25.

[44] Kong, A., Liu, J. S., and Wong, W. H. Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association 89*, 425 (1994), 278–288.

[45] Koski, T. *Hidden Markov Models for Bioinformatics*. Kluwer, 2001.

[46] Lee, A., and Whiteley, N. Forest resampling for distributed sequential Monte Carlo. *arXiv preprint* (2014).

[47] Lee, A., Yau, C., Giles, M., Doucet, A., and Holmes, C. C. On the utility of graphics cards to perform massively parallel implementation of advanced Monte Carlo methods. *Journal of Computational Graphical Statistics 19* (2010).

[48] Lindsten, F., Johansen, A. M., Naesseth, C. A., Kirkpatrick, B., Schon, T., Aston, J., and Bouchard, A. Divide-and-conquer with sequential Monte Carlo. *arXiv preprint* (2014).

[49] Liu, J. S., and Chen, R. Sequential Monte Carlo methods for dynamic systems. *Journals of the American Statistical Association 93*, 443 (1998), 1032–1044.

[50] Meyn, S., and Tweedie, R. L. *Markov Chains and Stochastic Stability. Second Edition*. Cambridge University Press, 2009.

[51] Persing, A., and Jasra, A. Likelihood computation for hidden Markov models via generalized two-filter smoothing. *Statistics and Probability Letters 83* (2013), 1433–1442.

[52] Pitt, M. K., dos Santos Silva, R., Giordani, P., and Kohn, R. On some properties of markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics 171*, 2 (2012), 134–151.

[53] Pitt, M. K., and Shephard, N. Filtering via simulation: auxiliary particle filter. *Journal of the American Statistical Association 94*, 446 (1999), 590–599.

[54] Rabiner, L. R., and Juang, B. H. *Fundamentals of Speech Recognition.* Prentice-Hall, 1993.

[55] Raftery, A. E., and Lewis, S. M. One long run with diagnostics: Implementation strategies for Markov chain Monte Carlo. *Statistical Science 7* (1992), 493–497.

[56] Raftery, A. E., and Lewis, S. M. The number of iterations, convergence diagnostics and generic metropolis algorithms. In *Practical Markov Chain Monte Carlo (W.R. Gilks, D.J. Spiegelhalter and S. Richardson, eds.* (1995), Chapman and Hall, pp. 115–130.

[57] Ristic, B., Arulampalam, M., and Gordon, A. *Beyond Kalman Filters: Particle Filters for Target Tracking.* Artech House, 2004.

[58] Robert, C. P., and Casella, G. *Monte Carlo statistical method 2nd Edition.* Springer, 2004.

[59] Roberts, G. O., Gelman, A., and Gilks, W. R. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability 7*, 1 (1997), 110–120.

[60] Roberts, G. O., and Smith, A. F. M. Simple conditions for the convergence of the gibbs sampler and Hastings-Metropolis algorithms. *Stochastic Processes and their applications 49* (1994), 207–216.

[61] Shao, J. *Mathematical Statistics 2nd Edition*. Springer, 2003.

[62] Shephard, N. Statistical aspects of ARCH and stochastic volatility. *Econometric Theory 27*, 5 (1996), 1–67.

[63] Vergé, C., Dubarry, C., Del Moral, P., and Moulines, E. On parallel implementation of sequential Monte Carlo methods: the island particle model. *Statistics and Computing 23* (2013), 1–18.

[64] Whiteley, N., Lee, A., and Heine, K. On the role of interaction in sequential Monte Carlo algorithms. *arXiv preprint* (2013).

[65] Yu, J. On leverage in a stochastic volatility model. *Journal of Econometrics 127* (2005), 165–178.